UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingenieros de Telecomunicación

TESIS DOCTORAL

INDUCCIÓN DE CONOCIMIENTO CON INCERTIDUMBRE EN BASES DE DATOS RELACIONALES BORROSAS

ANTONIO-JOSÉ GÓMEZ FLECHOSO

Ingeniero de Telecomunicación

Madrid, 1998

1998 TESIS DOCTORAL **ANTONIO-JOSÉ GÓMEZ FLECHOSO** 09

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS TELEMÁTICOS

Escuela Técnica Superior de Ingenieros de Telecomunicación

Universidad Politécnica de Madrid

INDUCCIÓN DE CONOCIMIENTO CON INCERTIDUMBRE EN BASES DE DATOS RELACIONALES BORROSAS

TESIS DOCTORAL

Autor

Antonio-José Gómez Flechoso

Ingeniero de Telecomunicación

Director

Gregorio Fernández Fernández

Doctor Ingeniero de Telecomunicación

Año: 1998

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día ..... de ..... de 19.....

Presidente D. .....

Vocal D. .....

Vocal D. .....

Vocal D. .....

Secretario D. .....

Realizado el acto de la defensa y lectura de la Tesis el día .....

de ..... de 19.....

en .....

Calificación: .....

EL PRESIDENTE **LOS VOCALES**

EL SECRETARIO

A mis padres y a Paloma

AGRADECIMIENTOS

Tengo que expresar mi agradecimiento, en primer lugar, a Gregorio Fernández quien, como Director de Tesis, considero que ha realizado un trabajo excelente. Siempre me ha dejado suficiente libertad como para dar mis propios pasos en un terreno tan novedoso y atractivo como la inducción de conocimiento en bases de datos y, sin embargo, ha sabido guiar mi camino de modo que nunca me sintiera perdido o desorientado. Sin duda, el haber tenido a Gregorio como director de tesis es una garantía de que el esfuerzo realizado no ha sido baldío.

Por otro lado, siguiendo en el ámbito académico, tengo que mencionar a Juan Ramón Velasco, siempre dispuesto a echarme una mano. Además debo agradecerle, especialmente, el apoyo que me dio para que no abandonara la Tesis cuando, por diferentes motivos, renuncié a la beca que me vinculaba a la Escuela.

También he de recordar aquí a José Carlos González, Mercedes Garijo, José Miguel Goñi, Fernando Sáez Vacas y otros del Grupo de Sistemas Inteligentes que, en un momento u otro, me han ayudado con los problemas con que me tropezaba. Especialmente, Amalio y Carlos Ángel, compañeros de fatiga con los que he compartido muchas experiencias a lo largo de estos años; su ayuda desde la Escuela ha sido decisiva en muchas ocasiones, por eso quiero desearles todo el éxito en sus respectivas tesis.

Ha habido más gente que, de un modo u otro también han intervenido en este trabajo: David Serrano, quien participó activamente durante el proyecto SEIC; Fairouz Medhajed, con quien he mantenido diferentes discusiones, todas provechosas; Pedro Alonso y Beatriz Buriel, que se animaron a aplicar FZFOIL en su proyecto de teletrabajo (ayudándome a depurar el código con sus pruebas), etc.

Es difícil mencionar a todos los que, en algún momento, me han animado a seguir. Por ejemplo, compañeros y amigos de la Escuela (Antonio de Lucas, Carlos García, Javi, Asen, Marco, etc.), amigos de otros círculos (Honorio, José Carlos, José Ramón y, sobre todo, Antonio, mi primer amigo doctor, y María Teresa), compañeros y amigos de Telefónica I+D (Toni, Chon, Gustavo, Nuria, y todos los demás, especialmente Juanan, que me ha ayudado siempre que lo he necesitado), ...

Pero el mayor apoyo lo he recibido siempre de mi familia. A ellos les debo todo lo que tengo. Siempre me han ayudado en lo que han podido y sé que lo volverían a hacer cuantas veces fuera necesario. Mis padres han sufrido conmigo todos los momentos difíciles y me han hecho llevadero, durante estos últimos años, el esfuerzo que supone invertir casi todo mi tiempo libre en una Tesis que, a veces, parecía interminable. También mis hermanas Mariángeles (ya doctora en Astrofísica) y Cristina (aún estudiante de Montes, pero que llegará lejos), han sido mis animadoras y han sufrido mis ratos de mal humor como yo no hubiera podido hacer. Paloma comenzó su relación conmigo a la vez que esta Tesis (por supuesto no la terminará a la vez) y creo que ella, primero como novia, ahora como esposa, es la que mejor conoce las dificultades y sacrificios de este camino. Sin ella hubiera sido imposible.

A todos vosotros os doy las gracias.

Índice de Contenidos

Índice de Contenidos (cont.)

Índice de Contenidos (cont.)

Índice de Contenidos (cont.)

Índice de Contenidos (cont.)

Índice de Contenidos (cont.)

Índice de Contenidos (cont.)

Resumen

Este trabajo presenta un sistema para aprendizaje de definiciones lógicas con incertidumbre, a partir de una base de datos relacional borrosa.

El campo de interés se centra, por tanto, en la programación lógica inductiva, introduciendo algunas interesantes aportaciones, principalmente en lo que se refiere a la entrada de datos y a los resultados producidos:

- Los datos de entrada pertenecen a una base de datos relacional borrosa. Por tanto, vienen expresados en forma de tablas de tuplas (relaciones), en las que las tuplas **pueden** llevar asociado un grado de pertenencia a la relación correspondiente. Se trata, por tanto, de relaciones borrosas, directamente identificables con conceptos borrosos (tan comunes en la realidad vista desde un punto de vista humano), y no de relaciones ordinarias con atributos borrosos (tal y como se entiende la "borrosidad" en muchos sistemas existentes).
- Los datos de salida vienen expresados en forma de definiciones lógicas de una relación (ordinaria o borrosa), que consta de una cláusula de Horn o de la disyunción de varias. Estas cláusulas de Horn se construyen mediante literales, aplicados sobre variables (generalmente), y asociados a relaciones borrosas u ordinarias. Los literales borrosos pueden ser modificados, además, por el empleo de etiquetas lingüísticas. Por tanto, se combina, en estas definiciones, la lógica de predicados con la lógica borrosa, en lo que podemos denominar "lógica borrosa de predicados", lo que constituye una aportación dentro de la inducción automática de conocimiento. Además, las definiciones inducidas llevan asociado un factor de incertidumbre, como hacen otros sistemas ya existentes.

El punto de partida del trabajo lo constituye un sistema de inducción de definiciones lógicas bien conocido: FOIL, creado por Quinlan en 1990, basado en la lógica de predicados. Sobre este sistema inicial se realizan, además de las extensiones para lógica borrosa ya mencionadas, otra serie de modificaciones y ampliaciones enfocadas a mejorar la inducción de conocimiento. Estas mejoras se realizan, principalmente, en su parte heurística, al definir una función de evaluación de literales, basada en medidas de interés, que permite corregir algunas deficiencias del sistema original y aumentar la calidad de las reglas inducidas. Otras modificaciones se orientan hacia la introducción de conocimiento de base, mediante relaciones definidas intensionalmente, de modo similar a otros sistemas como FOCL.

Como resultado tangible de la tesis, se ha desarrollado y probado un sistema, FZFOIL, disponible públicamente bajo la licencia GNU.

Abstract

This work shows a system to learn logical definitions with uncertainty, from a fuzzy relational database. This learning system is attractive by some new aspects in inductive logic programming, specially related with input data and generated results:

- Input data are a fuzzy relational database. Therefore, data are represented by tables (relations), whose tuples **are allowed** to have an associated membership grade. That means, relations are fuzzy, similar to human concepts (which introduce fuzziness as an usual characteristic of reality), and they are not crisp relations with fuzzy attributes (misunderstanding of "fuzziness" in many current systems).
- Output data are expressed as logical definitions of (fuzzy or crisp) relations, with one or several Horn clauses. These Horn clauses are constituted by literals, applied (usually) over variable attributes and associated to crisp or fuzzy relations. Fuzzy literals can be modified with linguistic hedges. In this way, first order logic and fuzzy logic are combined together to obtain a "fuzzy first order logic", novel contribution to automatic knowledge induction. Induced definitions have associated an uncertainty factor, like other existing systems.

The starting point of the work is a well known learning system, FOIL (Quinlan, 90), based on first order logic. From this initial system, several modifications are introduced, some of them are extensions to fuzzy logic, as mentioned, and some other are focused to improve the induction process. These improvements are oriented, mainly, to the heuristic components of FOIL, they are based on interest-based measures, and they solve some deficiencies on the initial system. Some other modifications are oriented to the introduction of background knowledge, using intensional relations like other systems (eg. FOCL).

As a result of the thesis, a new system, FZFOIL, has been developed and tested, and now can be reached on a public site, under GNU license.

.

ah

Capítulo 1:

INTRODUCCIÓN

Nos encontramos en la denominada “sociedad de la información”, porque se destinan gran cantidad de recursos a la adquisición, almacenamiento, procesado, análisis, etc. de la información. El conocimiento más valioso suele aparecer oculto entre los datos recogidos, en forma de patrones o reglas que relacionan entre sí otras partes más superficiales de la información. Este conocimiento se ha venido obteniendo, tradicionalmente, mediante análisis manual, aplicando la inferencia inductiva sobre el conjunto de datos de partida.

Sin embargo, la adquisición y almacenamiento de los datos se realiza a un ritmo cada vez mayor. Por ejemplo, los satélites de observación de la Tierra generarán, previsiblemente, del orden de un petabyte de datos (10^{15} bytes) diariamente a finales de siglo; otros sistemas menos sofisticados, como las transacciones realizadas en un supermercado, cabinas de información de turismo, operaciones de tarjetas de crédito, etc. también son susceptibles de generar un volumen de datos imposible de analizar de forma manual. La explosión en el número de fuentes de información disponibles en *Internet* ofrece una nueva oportunidad de búsqueda y extracción de información útil a partir de esta “base de datos” dinámica y creciente. Se estima que cada 20 meses se duplica la cantidad de información en el mundo. Parece claro que el clásico método hipotético-deductivo de la ciencia positiva resulta inoperante ante tal avalancha de datos, al menos, aplicado de la forma tradicional, esto es, analizando manualmente los datos disponibles. Cada vez se necesita más la ayuda de ordenadores potentes para automatizar el proceso inductivo, para analizar de forma inteligente las montañas de datos existentes, y extraer de ellas ese conocimiento oculto y valioso. Sin duda, el término “minería de datos”, con que se conocen las nuevas técnicas de análisis automático, refleja bastante bien esta idea.

Por otro lado, el esfuerzo inicial por adquirir conocimiento del mundo real está dejando paso a un mayor esfuerzo por conocer aspectos del propio conocimiento. Hoy en día nos encontramos este último punto, quizá como consecuencia de los fallos en el conocimiento adquirido mediante ese esfuerzo inicial. La preocupación principal ya no es la mera adquisición de conocimiento, sino la delimitación de su alcance y validez; necesitamos asignar un grado de certeza al conocimiento, saber en qué medida conocemos algo.

La incertidumbre está asociada, de forma inseparable, con la información. Aunque existen diferentes formas de incertidumbre, cabe destacar la que se produce como consecuencia de la imprecisión y subjetividad propias de la actividad humana. En muchas ocasiones sacrificamos parte de la información precisa disponible por otra más vaga pero más robusta, lo que permite

manejar de forma eficiente la complejidad asociada al mundo real. Muchos de los conceptos manejados habitualmente son, en sí mismos, vagos o borrosos, es decir, sus límites no están perfectamente determinados, y no por ello carecen de significación. La lógica borrosa y la teoría de conjuntos borrosos ofrecen un método natural para representar esa imprecisión y subjetividad humanas.

La combinación de técnicas de minería de datos con la incertidumbre asociada a los conceptos del mundo real, desde el punto de vista humano, nos conducen a la inducción automática de conocimiento con incertidumbre, tema sobre el que se desarrollará esta tesis

1.1 Justificación de esta tesis

Esta tesis se enmarca dentro del vasto campo de la inducción automática de conocimiento a partir de bases de datos. En particular, dentro del aprendizaje empírico de primer orden y la programación lógica inductiva, en gran auge desde hace unos 10 años.

Aunque muchos sistemas de aprendizaje basados en la lógica de proposiciones ya han demostrado su validez al aplicarlos sobre conjuntos de datos reales, en general, no ocurre lo mismo con los sistemas de programación lógica inductiva. La expresividad de la lógica de predicados permite construir descripciones sencillas, aunque dificulta enormemente la construcción de las mismas, al producirse una explosión combinatoria en el espacio de búsqueda. Esto hace que la búsqueda de la mejor descripción se convierta en una difícil y costosa tarea, sólo posible mediante la aplicación de métodos heurísticos, ya usados por la mayoría de estos sistemas.

La aplicación de sistemas de “laboratorio” para inducción de conocimiento a partir de bases de datos reales presenta dificultades adicionales, asociadas al ruido e incertidumbre de la realidad, al gran volumen de datos existente en una base de datos real, así como a la dinamicidad e incompletitud de algunos datos.

El trabajo central de esta tesis se orienta, precisamente, hacia la aplicación de un sistema de programación lógica inductiva bien conocido, FOIL ([Quinlan, 90]), sobre bases de datos reales.

Para ello, se analizará, en primer lugar, la parte heurística que utiliza, en concreto la función *Ganancia* de evaluación de literales, que presenta algunas deficiencias en situaciones concretas que pueden presentarse no sólo en la realidad sino también en pequeños ejemplos de entrenamiento. Este punto guarda una estrecha relación con un problema bien conocido en muchos sistemas de optimización: los máximos locales. Una sustitución de esta función de evaluación por otra basada en medidas del interés de los literales, según ciertos criterios, así como algunas modificaciones en la construcción de los conjuntos de entrenamiento intermedios, mejoran sustancialmente el comportamiento global del algoritmo, solventando los problemas detectados.

Por otro lado, se introducirá una importante mejora en el sistema anterior, enfocada hacia la manipulación de la incertidumbre asociada a la realidad. Se propone la utilización de la lógica borrosa y de conjuntos borrosos para representar, de un modo más natural, conceptos borrosos dentro de la base de datos relacional de partida. La adaptación del algoritmo para manejar relaciones borrosas a la entrada y construir definiciones con incertidumbre a la salida amplía las posibilidades y la expresividad (más afín a la concepción humana de la realidad) del sistema original. Esto supone una interesante aportación dentro de la programación lógica inductiva, ya que los sistemas existentes hasta ahora no van más allá de la manipulación de relaciones ordinarias (booleanas) o, a lo sumo, de relaciones en las que algunos atributos tienen asociado un

factor de incertidumbre (enfocados a la lógica de proposiciones extendida). El sistema que se propone, por el contrario, maneja relaciones borrosas en sí mismas (con incertidumbre asociada a sus tuplas) y las descripciones que construye se basan en la lógica borrosa y en la lógica de predicados. Es decir, este sistema podría decirse que utiliza la "lógica borrosa de primer orden" para representar el conocimiento inducido. Sobre este punto será necesario desarrollar algunos conceptos teóricos y definiciones nuevas.

1.2 Objetivos

El principal objetivo que se planteó al comenzar este trabajo fue la adaptación de un sistema de **programación lógica inductiva** para poder aplicarlo sobre **datos reales** y su extensión para poder manejar **conocimiento con incertidumbre**, tanto en los datos de entrada (en la base de datos relacional de partida) como en las definiciones lógicas que se induzcan.

Para conseguir este objetivo, se desglosó el mismo en diferentes subobjetivos:

1. Revisión de diferentes sistemas de aprendizaje basados en la programación lógica inductiva, así como de diferentes tipos de incertidumbre y de las teorías matemáticas adecuadas para su representación. Como resultado de este punto se seleccionó un sistema de aprendizaje de primer orden (FOIL), por su flexibilidad para representar el conocimiento, frente a otros métodos de representación del mismo. Por otro lado, se eligió un modo de representar la incertidumbre en la información (lógica borrosa), adecuado para el tipo de conocimiento que se maneja habitualmente, desde un punto de vista humano. A partir de ambas selecciones se planteó la construcción de un único sistema que incorporase las mejores ventajas de ambos: potencia expresiva de la lógica de primer orden y utilidad de la representación de la incertidumbre.

2. El algoritmo inicial, FOIL, se analizó y recodificó en C++ para facilitar su mantenimiento y la incorporación de mejoras en algunos aspectos en los que presenta deficiencias, principalmente en su parte heurística. Con estas modificaciones se pretendía corregir algunos problemas que pueden presentarse en bases de datos reales, y que conducen a la no construcción de definiciones o a la construcción de definiciones más complejas de lo necesario. Asimismo se realizaron otras pequeñas modificaciones para facilitar el acceso a grandes volúmenes de datos (muestreo en el conjunto de entrenamiento), introducción de conocimiento de base (relaciones intensionales), etc.

3. Extensiones orientadas hacia la lógica borrosa de primer orden. En este punto, cabe adelantar la no existencia de un formalismo teórico que incorpore de forma simultánea la lógica de predicados y la lógica borrosa, por lo que ha sido necesario redefinir diferentes aspectos teóricos de la lógica de predicados, para asentar las bases de una lógica borrosa de primer orden. Por otro lado, se amplió el formato de la base de datos de entrada para permitir, además de las relaciones ordinarias, la definición de relaciones borrosas, en el sentido estricto de dicho término: relaciones en las que sus tuplas incorporan un factor de incertidumbre, en forma de grado de pertenencia a la relación. Y, finalmente, se modificaron el algoritmo de partida y sus heurísticos para que se pudiera manejar esta información y se permitiera construir definiciones borrosas (con literales borrosos), tanto para relaciones ordinarias como borrosas. Las definiciones que se inducen siguen incorporando un factor de incertidumbre, similar a una medida de probabilidad que, del mismo modo que en FOIL, da una idea de su precisión, para poder ser aplicadas en un sistema experto.

1.3 Estructura de la tesis

Cada uno de los objetivos expuestos en el apartado anterior será tratado en un capítulo diferente de la tesis:

- En el capítulo 2 se presenta el estado del arte en descubrimiento de conocimiento en bases de datos y minería de datos, con especial hincapié en los sistemas de programación lógica inductiva. Dentro de estos últimos, se selecciona el sistema FOIL, del que se realiza una descripción detallada, tanto de su parte algorítmica como de su parte heurística; esta parte incluye una reformulación del sistema, respecto de la descripción original realizada por Quinlan, a partir de una serie de conceptos útiles que se definen inicialmente. Por último, este capítulo incluye una descripción de diferentes tipos de incertidumbre y de las teorías matemáticas con las que se abordan; como resultado se justifica la selección de la lógica borrosa para la representación de la incertidumbre en nuestro sistema.
- En el capítulo 3 se analizan diferentes problemas que surgen en la versión original de FOIL, derivados de un deficiente funcionamiento de la función *ganancia* de evaluación de literales. Se propone un nuevo heurístico de evaluación, basado en medidas de interés, y una modificación en el conjunto de entrenamiento a considerar durante la construcción de una cláusula, con lo que se mejora sustancialmente el comportamiento del sistema en ejemplos concretos. Finalmente se incluyen un par de ejemplos comparativos, que permiten evaluar la importancia de los cambios realizados.
- En el capítulo 4 se amplían diferentes aspectos teóricos de la lógica de predicados, extendiéndolos para su utilización en "lógica borrosa de predicados". A continuación se realiza una descripción del nuevo sistema, FZFOIL, siguiendo una línea paralela a la seguida para FOIL: definición de una serie de conceptos iniciales y formulación del algoritmo a partir de ellos. Por último se tratan temas como la utilización de relaciones intensionales o las etiquetas lingüísticas en los literales borrosos.

  Para la descripción de FZFOIL se utilizan diferentes conceptos de la lógica borrosa que, a modo de resumen, se recopilan en el apéndice A: operadores en conjuntos borrosos, relaciones borrosas y proposiciones borrosas.

A continuación (capítulo 5) se presentan y analizan algunos resultados obtenidos, tras la aplicación del algoritmo sobre una base de datos real, procedente del proyecto SEIC (PC-183, dentro de la línea de "Acciones Especiales PASO"). En este capítulo, además, se plantean otros puntos de interés, como el análisis de la complejidad del algoritmo (algunas expresiones matemáticas utilizadas en este punto se desarrollan en el apéndice B). En el apéndice C se incluye un pequeño manual de usuario y de instalación de la versión de FZFOIL implementada, con la que se han obtenido los resultados; este manual describe el formato de los datos de entrada y las opciones y órdenes seleccionables por el usuario, así como instrucciones sobre dónde conseguir el programa y cómo instalarlo.

Finalmente, el capítulo 6 recopila diferentes conclusiones y propone posibles líneas de investigación futuras.

Capítulo 2:

ESTADO DEL ARTE

2.1 Introducción

En muchas áreas del saber, el conocimiento se ha venido obteniendo por el clásico método hipotético-deductivo de la ciencia positiva. En él es fundamental el paso inductivo inicial: a partir de un conjunto de observaciones y de unos conocimientos previos, la intuición del investigador le conduce a formular la hipótesis. Esta "intuición" resulta inoperante cuando no se trata de observaciones aisladas y casuales, sino de millones de datos almacenados en soporte informático. En el fondo de todas las investigaciones sobre inducción en bases de datos subyace la idea de automatizar ese paso inductivo.

Las técnicas de análisis estadístico, desarrolladas hace tiempo, permiten obtener ciertas informaciones útiles, pero no inducir relaciones cualitativas generales, o *leyes*, previamente desconocidas; para esto se requieren técnicas de análisis inteligente ([Frawley et al., 91]) que todavía no han sido perfectamente establecidas. Por ello, se incrementa de forma continua la diferencia existente entre la cantidad de datos disponibles y el conocimiento extraído de los mismos. Pero cada vez más investigaciones dentro de la inteligencia artificial están enfocadas a la inducción de conocimiento en bases de datos. Consecuencia de esta creciente necesidad ha aparecido un nuevo campo de interés: la *minería de datos* (*data mining*), que incluye los nuevos métodos matemáticos y técnicas software para análisis inteligente de datos. La minería de datos surge a partir de sistemas de aprendizaje inductivo en ordenadores, al ser aplicados a bases de datos ([Holsheimer y Siebes, 94]), y su importancia crece de tal forma que incluso es posible que, en el futuro, los sistemas de aprendizaje se usen de forma masiva como herramientas para analizar datos a gran escala.

Se denomina *descubrimiento de conocimiento en bases de datos* (*KDD*) al proceso global de búsqueda de nuevo conocimiento a partir de los datos de una base de datos. Este proceso incluye no sólo el análisis inteligente de los datos con técnicas de *minería de datos*, sino también los pasos previos, como el filtrado y preprocesado de los datos, y los posteriores, como la interpretación y validación del conocimiento extraído.

Normalmente el término *minería de datos* lo usan estadísticos, analistas de datos, y la comunidad de sistemas de gestión de información, mientras que *KDD* es más utilizado en inteligencia artificial y aprendizaje en ordenadores.

2.2 Descubrimiento de conocimiento y minería de datos

El término *descubrimiento de conocimiento en bases de datos* (*knowledge discovery in databases*, o *KDD* para abreviar) empezó a utilizarse en 1989[1] para referirse al amplio proceso de búsqueda de conocimiento en bases de datos, y para enfatizar la aplicación a "alto nivel" de métodos específicos de minería de datos ([Fayyad et al., 96b]).

En general, el *descubrimiento* es un tipo de inducción de conocimiento, no supervisado ([Michalski, 87]), que implica dos procesos:

- búsqueda de regularidades interesantes entre los datos de partida,
- formulación de *leyes* que las describan.

Entre la literatura dedicada al tema, se pueden encontrar varias definiciones para *descubrimiento*:

- Según [Walker, 87], el descubrimiento implica observar, recoger datos, formar hipótesis para explicar las observaciones, diseñar experimentos, comprobar la corrección de las hipótesis, comparar nuestros hallazgos con los de otros investigadores y repetir el ciclo. Los ordenadores son capaces de observar y recoger datos, a veces mejor que los observadores humanos; los programas estadísticos pueden generar agrupaciones de forma automática entre los datos recogidos, aunque no siempre se corresponden con las clasificaciones hechas por los hombres; también hay programas con cierta capacidad para diseñar experimentos; y algunos sistemas robóticos realizan las manipulaciones necesarias en ciertos experimentos. Pero ningún ordenador reúne todas estas habilidades ni es capaz de adaptarse para aplicarlas a nuevos problemas; en este sentido, los ordenadores no serían capaces de descubrir. Sin embargo, el descubrimiento no requiere realizar simultáneamente todas estas tareas. De igual modo que un investigador puede descubrir nuevo conocimiento a través del análisis de sus datos, un ordenador puede examinar los datos disponibles o recogidos por otros ordenadores y encontrar relaciones y explicaciones previamente desconocidas, realizando así descubrimiento en un sentido más restringido. La capacidad de los ordenadores para realizar búsquedas exhaustivas de forma incansable entre grandes cantidades de datos ofrece buenas expectativas para obtener descubrimiento de forma automática.
- Según [Frawley et al., 91] el *descubrimiento de conocimiento* es la extracción no trivial de información implícita, previamente desconocida y potencialmente útil, a partir de un conjunto de datos. Dado un conjunto de hechos (datos) H, un lenguaje L, y alguna medida de la certidumbre C, definimos una regularidad (*pattern*) como una sentencia S en L que describe relaciones dentro de un subconjunto H_S de H con una certidumbre c, de forma que S es más sencillo que la enumeración de todos los hechos de H_S. Una regularidad que sea interesante y bastante cierta (según criterios definidos por el usuario) se denomina *conocimiento*. Un sistema de descubrimiento será un programa que toma como entrada el conjunto de hechos y extrae las regularidades existentes. Cuando el conocimiento se extrae partiendo de los datos de una base de datos, se tiene *KDD*.

1. Según indica [Fayyad, 96], la comunidad de KDD ha crecido de forma significativa en los últimos años, como muestra el creciente número de congresos organizados sobre este tema. La primera de estas reuniones (*workshop*) tuvo lugar en 1989, surgiendo a partir de ella el primer libro: [Piatetsky-Shapiro y Frawley, 91]. Hubo otras reuniones posteriores hasta la de 1994 (*KDD-94 workshop*) con 80 participantes, a partir de la cual se cambió el formato de las mismas para permitir la asistencia de público. En la última de ellas (*Second International Conference on Knowledge Discovery and Data Mining o KDD-96*) hubo unos 500 participantes.

Los conceptos de lenguaje, certeza, simplicidad e interés, con los que se define el descubrimiento de conocimiento, son lo suficientemente vagos como para que esta definición cubra una amplia variedad de tendencias. Sin embargo, son ideas fundamentales que diferencian el KDD de otros sistemas de aprendizaje ([Frawley et al., 91]):

- Lenguaje de alto nivel: El conocimiento descubierto se representa en un lenguaje de alto nivel, inteligible desde el punto de vista humano. Por tanto, quedan descartadas, dentro del KDD, representaciones de bajo nivel como las generadas por redes neuronales (a pesar de que éstas son un método válido de minería de datos).
- Precisión: Los descubrimientos representan el contenido de la base de datos que, como reflejo de la realidad, puede contener imperfecciones y ruido. Por tanto, será raro que algún conocimiento se cumpla con todos los datos. El grado de certidumbre medirá el crédito o confianza que el sistema o usuario puede asignar a cierto descubrimiento; si la certeza no es lo suficientemente alta, los patrones descubiertos no llegarán a ser conocimiento.
- Interés: Aunque es posible extraer numerosos patrones de cualquier base de datos, sólo se consideran como conocimiento aquéllos que resulten interesantes según ciertos criterios del usuario. En particular, un patrón interesante debe ser nuevo, potencialmente útil y no trivial.
- Eficiencia: Son deseables procesos de descubrimiento que puedan ser eficientemente implementados en un ordenador. Se considera que un algoritmo es eficiente cuando su tiempo de ejecución y el espacio de memoria requerido crecen de forma polinómica con el tamaño de los datos de entrada. No es posible aprender de forma eficiente cualquier concepto booleano (problema NP-completo), sin embargo, sí existen algoritmos eficientes para clases restringidas de conceptos, como los representables en forma conjuntiva, etc. Otra posibilidad es el uso de heurísticos y algoritmos aproximados para la inducción de conocimiento.

Cualquier algoritmo usado en un proceso de KDD debe considerar que *conocimiento* es una sentencia S expresada en un lenguaje L, cuyo interés I (según criterios del usuario) supera cierto umbral i (definido también por el usuario). A su vez, el interés depende de criterios de certeza, simplicidad y utilidad, establecidos por el usuario. Según se definan los umbrales para estos criterios, se puede enfatizar la búsqueda de información precisa (gran certeza), o útil, etc.

En [Brachman y Anand, 96] se define el proceso de KDD, desde un punto de vista práctico, como “una tarea intensiva en conocimiento que consta de complejas interacciones, prolongadas en el tiempo, entre un humano y una (gran) base de datos, posiblemente con la ayuda de un conjunto heterogéneo de herramientas”.

Los principales pasos dentro del proceso interactivo e iterativo del KDD pueden verse en la figura 2-1, y son los siguientes ([Fayyad et al., 96b] y [Fayyad, 96]):

1. Desarrollo y entendimiento del dominio de la aplicación, el conocimiento relevante y los objetivos del usuario final. Este paso requiere cierta dependencia usuario/analista, pues intervienen factores como: conocer los cuellos de botella del dominio, saber qué partes son susceptibles de un procesado automático y cuáles no, cuáles son los objetivos, los criterios de rendimiento exigibles, para qué se usarán los resultados que se obtengan, compromisos entre simplicidad y precisión del conocimiento extraído, etc.

2. Creación del conjunto de *datos objetivo*, seleccionando el subconjunto de variables o ejemplos sobre los que se realizará el descubrimiento. Esto implica consideraciones sobre la homogeneidad de los datos, su variación a lo largo del tiempo, estrategia de muestreo, grados de libertad, etc.

3. *Preprocesado* de los datos: eliminación de ruido, estrategias para manejar valores ausentes, normalización de los datos, etc.

4. *Transformación y reducción* de los datos. Incluye la búsqueda de características útiles de los datos según sea el objetivo final, la reducción del número de variables y la proyección de los datos sobre espacios de búsqueda en los que sea más fácil encontrar una solución. Este es un paso crítico dentro del proceso global, que requiere un buen conocimiento del problema y una buena intuición, y que, con frecuencia, marca la diferencia entre el éxito o fracaso de la minería de datos (paso 7).

5. Elección del tipo de sistema para minería de datos. Esto depende de si el objetivo del proceso de KDD es la clasificación, regresión, agrupamiento de conceptos (*clustering*), detección de desviaciones, etc. (en [Fayyad et al., 96] pueden verse en detalle los diferentes métodos de minería de datos).

6. Elección del algoritmo de minería de datos.

7. *Minería de datos*. En este paso se realiza la búsqueda de conocimiento con una determinada representación del mismo. El éxito de la minería de datos depende en gran parte de la correcta realización de los pasos previos, por parte del usuario.

8. *Interpretación* del conocimiento extraído, con posibilidad de iterar de nuevo desde el primer paso. La obtención de resultados aceptables dependerá de factores como: definición de medidas del interés del conocimiento (de tipo estadístico, en función de su sencillez, etc.) que permitan filtrarlo de forma automática, existencia de técnicas de visualización para facilitar la valoración de los resultados o búsqueda manual de conocimiento útil entre los resultados obtenidos.

9. Consolidación del conocimiento descubierto, incorporándolo al sistema, o simplemente documentándolo y enviándolo a la parte interesada. Este paso incluye la revisión y resolución de posibles inconsistencias con otro conocimiento extraído previamente.

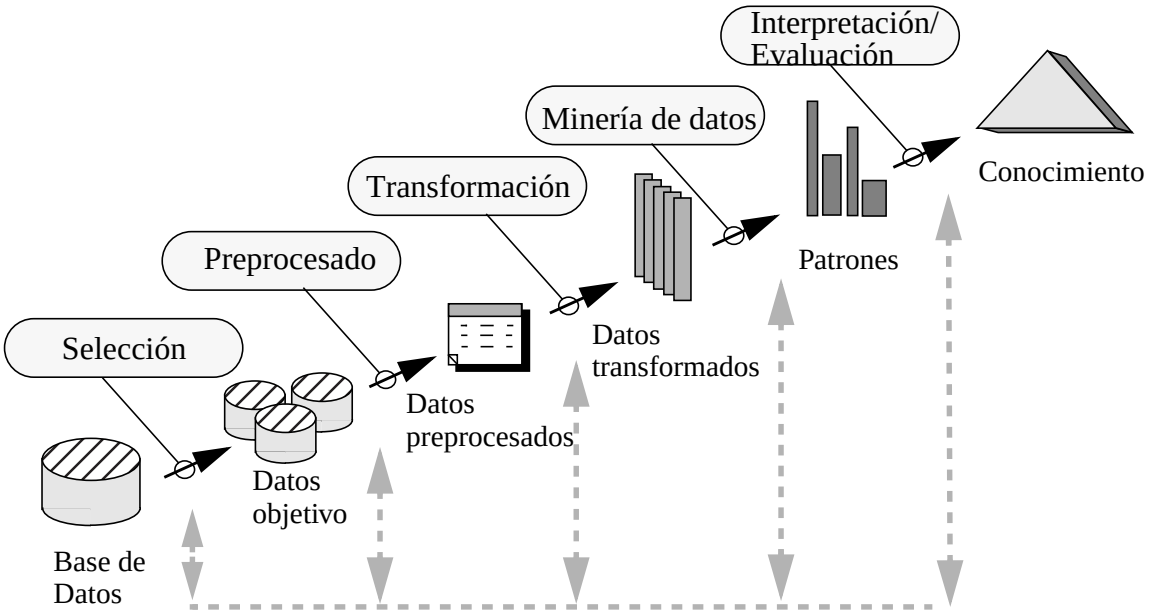

Figura 2-1: Proceso de KDD

Muchas veces los pasos que constituyen el proceso de KDD no están tan claramente diferenciados como se muestra en la figura anterior. Las interacciones entre las decisiones tomadas en diferentes pasos, así como los parámetros de los métodos utilizados y la forma de representar el problema suelen ser extremadamente complejos. Pequeños cambios en una parte pueden afectar fuertemente al resto del proceso.

Sin quitar importancia a ninguno de estos pasos del proceso de KDD, se puede decir que la minería de los datos es la parte fundamental, en la que más esfuerzos se han realizado.

Históricamente, el desarrollo de la estadística nos ha proporcionado métodos para analizar datos y encontrar correlaciones y dependencias entre ellos. Sin embargo, el análisis de datos ha cambiado recientemente y ha adquirido una mayor importancia, debido principalmente a tres factores ([Decker y Focardi, 95]):

a) Incremento de la potencia de los ordenadores. Aunque la mayoría de los métodos matemáticos fueron desarrollados durante los años 60 y 70, la potencia de cálculo de los grandes ordenadores de aquella época (equivalente a la de los ordenadores personales de hoy en día) restringía su aplicación a pequeños ejemplos “de juguete”, fuera de los cuales los resultados resultaban demasiado pobres. Algo similar ha ocurrido con la capacidad de almacenamiento de los datos y su coste asociado.

b) Incremento del ritmo de adquisición de datos. El crecimiento de la cantidad de datos almacenados se ve favorecido no sólo por el abaratamiento de los discos y sistemas de almacenamiento masivo, sino también por la automatización de muchos experimentos y técnicas de recogida de datos. Se estima ([Frawley et al., 91]) que la cantidad de información almacenada en todo el mundo se duplica cada 20 meses; el número y tamaño de las bases de datos probablemente crece más rápidamente. Por ejemplo, se espera que los satélites de observación de la Tierra generen, a final de siglo, aproximadamente un *petabyte* (10^{15} *bytes*) de datos diariamente, por lo que una persona trabajando 24 horas al día, todos los días del año, a un ritmo de procesamiento de una imagen por segundo, necesitaría varios años para mirar las imágenes generadas en sólo un día.

c) Por último, han surgido nuevos métodos, principalmente de aprendizaje y representación de conocimiento, desarrollados por la comunidad de inteligencia artificial, estadística y física de dinámicas no lineales. Estos métodos complementan a las tradicionales técnicas estadísticas en el sentido de que son capaces de inducir relaciones cualitativas generales, o *leyes*, previamente desconocidas.

Estos nuevos métodos matemáticos y técnicas software, para análisis inteligente de datos y búsqueda de regularidades en los mismos, se denominan actualmente técnicas de *minería de datos* o *data mining*. A su vez, la minería de datos ha permitido el rápido desarrollo de lo que se conoce como *descubrimiento de conocimiento en bases de datos*.

Las técnicas de *minería de datos* han surgido a partir de sistemas de *aprendizaje inductivo en ordenadores*, siendo la principal diferencia entre ellos los datos sobre los que se realiza la búsqueda de nuevo conocimiento ([Holsheimer y Siebes, 94]). En el caso tradicional de aprendizaje en ordenadores (*machine learning*), se usa un conjunto de datos pequeño y cuidadosamente seleccionado para entrenar al sistema. Por el contrario, en la minería de datos se

parte de una base de datos[1], generalmente grande, en la que los datos han sido generados y almacenados para propósitos diferentes del aprendizaje con los mismos.

2.2.1 Limitaciones del aprendizaje sobre bases de datos

Podría parecer que una simple traducción de términos es suficiente para poder aplicar sobre las bases de datos técnicas tradicionales de aprendizaje en ordenadores (tabla 2-1), realizando el aprendizaje sobre las tuplas de la base de datos en vez de sobre el conjunto de ejemplos. Sin embargo, la inducción de conocimiento en bases de datos se encuentra con dificultades no previstas por los tradicionales sistemas de aprendizaje, pues en el mundo real, las bases de datos suelen ser dinámicas, incompletas, ruidosas y muy grandes ([Frawley et al., 91]).

Tabla 2-1: Terminología de bases de datos y sistemas de aprendizaje

Gestión de Bases de Datos	Sistemas de aprendizaje
Base de datos = colección ficheros dinámicos, lógicamente integrados	Base de datos = conjunto fijo de ejemplos
Fichero	Conjunto de datos o ejemplos
Tupla	Ejemplo, vector de características
Campo, atributo	Característica, atributo
Dominio de un atributo	Valores posibles de características
Diccionario de datos	Tipo de atributo e información de dominio
Datos relacionales	Conjunto de ejemplos
Condición lógica	Descripción de un concepto

Por estos motivos, la mayoría de los algoritmos de aprendizaje son ineficientes al ser aplicados sobre bases de datos y gran parte del trabajo que se realiza en la inducción de conocimiento en bases de datos trata de solucionar estos problemas:

a) Datos dinámicos

1. En terminología de bases de datos, se conoce por *base de datos* al conjunto de datos lógicamente integrados, que pueden pertenecer a uno o más ficheros y residir en uno o varios nodos. En una base de datos relacional, los datos están organizados en tablas, con tuplas de tamaño fijo. El almacenamiento, actualización y recuperación de los datos la realiza el *sistema gestor de la base de datos*, usando información contenida en el *diccionario de datos*, como nombre de los atributos, dominios de los mismos, etc.
Por el contrario, en aprendizaje en ordenadores y, en general, en inteligencia artificial ([Nilsson, 82]) se entiende por *base de datos* una colección de ejemplos almacenados en un único fichero. Estos ejemplos son generalmente vectores de características de longitud fija. En ocasiones se dispone de información sobre los nombres de las características y sus rangos de valores, como si de un diccionario de datos se tratara.

En la mayoría de las bases de datos, los datos son modificados de forma continua. Cuando el valor de los datos almacenados es función del tiempo, el conocimiento inducido varía según el instante en que se obtenga, y por ello es deseable un sistema que funcione de forma continua, en modo *secuencial*, para tener siempre actualizado el conocimiento extraído.

b) Datos incompletos

El manejo de datos incompletos en una base de datos puede deberse a pérdida de valores de algún atributo (al que se asigna entonces el valor *desconocido*), o a la ausencia del mismo en la vista que el sistema posee sobre los datos. En ambos casos, la incidencia en el resultado dependerá de si el dato incompleto es relevante o no para el objetivo del sistema de aprendizaje. Por ejemplo, un sistema para aprender a diagnosticar arritmias cardíacas no se verá afectado por la pérdida de datos como el color del pelo del paciente, pero sí por otros como el ritmo cardíaco.

c) Ruido e incertidumbre

El ruido existente en los datos viene determinado tanto por el tipo de valores de los atributos: real (ej. presión arterial), entero (ritmo cardíaco), cadena de caracteres (nombre del paciente) o nominal (tipo de arritmia), como por la exactitud en la medida de dichos valores (especialmente para atributos numéricos).

Por otro lado, es frecuente que las regularidades que se encuentran entre los datos de partida no se verifiquen siempre, sino con cierta probabilidad. Este efecto se debe no sólo a la presencia de ruido en la base de datos, sino también a la indeterminación existente en muchos aspectos de la realidad y a imprecisiones en el modelado de la misma (no hay que olvidar que en el diseño de la base de datos se modela la realidad, y todo modelo no es sino una aproximación más o menos precisa).

d) Tamaño de las bases de datos

El tamaño de las bases de datos suele ser muy superior al de los conjuntos de entrenamiento de muchos sistemas de aprendizaje, por ello, en las bases de datos muy grandes es inabordable un análisis completo de todos los datos, y deben emplearse técnicas específicas que aceleren el aprendizaje sobre las mismas. Esto marca una importante diferencia entre los sistemas de inducción aplicables en uno y otro caso:

- En primer lugar, a la hora de elegir un algoritmo para inducir conocimiento en grandes bases de datos, hay que considerar su eficiencia. La complejidad algorítmica debe ser la menor posible, para que los requerimientos de tiempos de computación no crezcan más que de forma polinómica, con un pequeño grado, al aumentar el tamaño de la base de datos.
- Además, algunos sistemas (como FOCL en [Pazzani y Kibler, 92], o SUBDUE en [Cook et al., 96]) utilizan conocimiento previo (*background knowledge* o *domain knowledge*) para guiar y limitar la búsqueda de conocimiento, mejorando la eficiencia. Sin embargo, la selección del conocimiento previo que va a emplearse ha de hacerse son sumo cuidado, pues podría descartar descubrimientos válidos no previstos, como se indica en [Piatetsky-Shapiro, 91] con un sencillo ejemplo: una restricción como “los camiones no circulan sobre el agua” reduce el espacio de búsqueda, pero limita algunas soluciones posibles como que los camiones vayan sobre lagos helados en invierno. El diccionario de datos es la forma más simple de conocimiento previo disponible; otra posibilidad es que sea proporcionado por un experto, o incluso que haya sido descubierto previamente por el propio sistema.

- Otra posibilidad para aumentar la eficiencia del sistema es utilizar algún tipo de heurístico que oriente la búsqueda de nuevo conocimiento, evitando realizar búsquedas exhaustivas.
- Por último, los algoritmos de descubrimiento pueden realizar algún tipo de muestreo dentro de la base de datos, para trabajar con una muestra del total. De este modo puede abordarse el análisis de bases de datos muy grandes, a costa de introducir incertidumbre en los resultados de la inducción.

La viabilidad del aprendizaje en bases de datos ya ha empezado a ser reconocida por algunos gobiernos y empresas, como indica [Piatetsky-Shapiro, 91], a pesar de que todavía es un tema que presenta algunas limitaciones. Por ejemplo, algunos bancos analizan bases de datos sobre créditos para encontrar los mejores criterios para su concesión o predecir situaciones de bancarrota. Otro ejemplo es la empresa General Motors, que construye de forma automática sistemas expertos para diagnosticar averías, partiendo para ello de bases de datos que recogen síntomas en los vehículos y problemas detectados. Cada vez se encuentran más referencias bibliográficas sobre aplicaciones prácticas de la minería de datos, como puede verse en [Decker y Focardi, 95], [Shortland y Scarfe, 94], [Mesrobian et al., 96], [John et al., 96], etc.

2.2.2 Disciplinas relacionadas

Por definición, se considera el KDD como un campo interdisciplinario en el que se reúnen investigadores de diferentes campos ([Frawley et al., 91], [Fayyad, 96]). A continuación veremos cómo están relacionados cada uno de ellos con las distintas partes del proceso de KDD:

- Estadística.

  Aunque la estadística proporciona una valiosa ayuda en el análisis de datos, generalmente no es suficiente, y presenta algunos inconvenientes: mala adecuación a datos de tipo nominal, sus resultados pueden ser difíciles de interpretar, y requiere que el usuario decida dónde y cómo analizar los datos. Sin embargo, juega un importante papel en algunos pasos dentro del proceso de KDD, sobre todo en la selección y muestreo de datos, en la minería de datos y en la evaluación del conocimiento extraído.

- Reconocimiento de patrones, aprendizaje en ordenadores e inteligencia artificial.

  Las técnicas de reconocimiento de patrones, especialmente las utilizadas para agrupamiento de conceptos (*clustering*) han contribuido en el proceso de KDD, sobre todo en la minería de datos. Estas técnicas utilizan estructuras de datos más sofisticadas que las técnicas estadísticas, a la vez que métodos más completos de búsqueda de conocimiento.

  Las técnicas procedentes de la inteligencia artificial se centran casi exclusivamente en el manejo de datos de tipo simbólico, aportando algoritmos basados en la búsqueda heurística y en modelos no paramétricos.

  Dentro de la inteligencia artificial, hay que destacar la importancia del aprendizaje en los ordenadores, sobre todo en la minería de datos, por sus técnicas para descubrir leyes, que describen relaciones cualitativas existentes en los datos. Otras áreas de la inteligencia artificial, como las relacionadas con el manejo de incertidumbre (por ej. redes bayesianas), representación del conocimiento o búsqueda, también tienen importancia en otros pasos dentro del KDD, como la transformación, selección y preprocesado de datos.

- Bases de datos y almacenes de datos (*data warehouses*).

Un sistema gestor de datos es un programa que permite el almacenamiento, acceso y modificación de los datos almacenados en una base de datos. Mediante el lenguaje de manipulación de datos permite realizar de forma sencilla algunas operaciones, como la selección de tuplas que cumplen cierta condición, pudiendo dar lugar a inducir reglas interesantes, por ejemplo "el número de accidentes de tráfico ha crecido linealmente en los últimos 5 años". Sin embargo, es el usuario el que decide qué operaciones realizar en cada momento sobre los datos, y el que induce la regla ante la respuesta obtenida. Las bases de datos deductivas ([Leung y Lee, 88]) y orientadas a objetos proporcionan nuevas posibilidades de análisis inteligente de los datos e inducción de conocimiento ([Silverschatz et al., 95]).

Los almacenes de datos o *warehouses* son una de las nuevas áreas de interés dentro de las bases de datos. No son más que copias de los datos de una o varias bases de datos, con integración de esquemas, que proporcionan un acceso a los mismos de una forma sencilla y uniforme.

2.3 Métodos aplicados de minería de datos

Pueden emplearse diferentes criterios para clasificar los sistemas de minería de datos y, en general, los sistemas de aprendizaje inductivo en ordenadores:

- Dependiendo del objetivo para el que se realiza el aprendizaje ([Fayyad et al., 96b]), pueden distinguirse sistemas para: clasificación (clasificar datos en clases predefinidas), regresión (función que convierte datos en valores de una función de predicción), agrupamiento de conceptos (búsqueda de conjuntos en los que agrupar los datos), compactación (búsqueda de descripciones más compactas de los datos), modelado de dependencias (dependencias entre las variables de los datos), detección de desviaciones (búsqueda de desviaciones importantes de los datos respecto de valores anteriores o medios), etc.
- Dependiendo de la tendencia con que se aborde el problema, se pueden distinguir tres grandes líneas de investigación o paradigmas: sistemas *conexionistas* (redes neuronales), sistemas *evolucionistas* (algoritmos genéticos) y sistemas *simbólicos*.
- Dependiendo del lenguaje utilizado para representar del conocimiento, se pueden distinguir: representaciones basadas en la lógica de proposiciones, representaciones basadas en lógica de predicados de primer orden, representaciones estructuradas, representaciones a través de ejemplos y representaciones no simbólicas como las redes neuronales.

A continuación describiremos con más detalle los diferentes métodos de representación del conocimiento que se emplean en la minería de datos, dado que el lenguaje de representación es uno de los aspectos importantes para el proceso de KDD.

También veremos otro de los aspectos importantes para la minería de datos: el aprendizaje en los ordenadores, del que ha heredado las técnicas para extraer conocimiento a partir de los datos.

2.3.1 Representación del conocimiento

2.3.1.1 Representaciones basadas en la lógica de proposiciones extendida

Los tradicionales sistemas de aprendizaje han utilizado con gran asiduidad, para representar el conocimiento, una extensión de la lógica de proposiciones, denominada lógica "0+" o representación objeto-atributo-valor. Dentro de la misma, pueden englobarse métodos de

representación equivalentes como los árboles de decisión, las reglas de producción y las listas de decisión:

a) Árboles de decisión

Los *árboles de decisión* son una forma de representación sencilla, muy usada entre los sistemas de aprendizaje supervisado, para clasificar ejemplos en un número finito de clases. Se basan en la partición del conjunto de ejemplos según ciertas condiciones que se aplican a los valores de los atributos. Su potencia descriptiva viene limitada por las condiciones o reglas con las que se divide el conjunto de entrenamiento; por ejemplo, estas reglas pueden ser simplemente relaciones de igualdad entre un atributo y un valor, o relaciones de comparación ("mayor que", etc.), etc.

Los sistemas basados en árboles de decisión forman una familia llamada TDIDT (*Top-Down Induction of Decision Trees*), cuyo representante más conocido es ID3 ([Quinlan, 86]).

ID3 (Interactive Dichotomizer) se basa en la reducción de la entropía media para seleccionar el atributo que genera cada partición (cada nodo del árbol), seleccionando aquél con el que la reducción es máxima. Los nodos del árbol están etiquetados con nombres de atributos, las ramas con los posibles valores del atributo, y las hojas con las diferentes clases. Existen versiones *secuenciales* de ID3, como ID5R ([Utgoff, 89]).

C4.5 ([Quinlan, 93], [Quinlan, 96]) es una variante de ID3, que permite clasificar ejemplos con atributos que toman valores continuos.

b) Reglas de producción

Una desventaja de los árboles de decisión es que tienden a ser demasiado grandes en aplicaciones reales y, por tanto, se hacen difíciles de interpretar desde el punto de vista humano. Por ello, se han realizado diversos intentos para convertir los árboles de decisión en otras formas de representación, como las *reglas de producción*. Aquí consideramos reglas de producción del tipo *si-entonces*, basadas en lógica de proposiciones. El consecuente es una clase, y el antecedente es una expresión proposicional, que puede estar en forma normal conjuntiva (CNF) o ser simplemente un término.

En [Quinlan, 87] se propone una técnica para construir reglas de decisión, basadas en lógica de proposiciones, a partir de árboles de decisión. El problema es que incluso las reglas de producción así obtenidas pueden resultar demasiado complejas para un experto humano.

Algunos sistemas como PRISM, descrito en [Cendrowska, 88], generan directamente reglas algo más sencillas, sin tener que construir el árbol previamente, mediante un algoritmo parecido a ID3 y con una precisión similar.

La familia AQ ([Michalski, 87]) la forman sistemas (AQ11, AQ15, etc.) que generan *descripciones estructurales*, por diferenciarlas de las *descripciones de atributos* de los sistemas anteriormente mencionados. Estas descripciones son también reglas de producción, aunque con mayor capacidad descriptiva, pues su antecedente es una fórmula lógica. La notación utilizada en estas reglas se denomina VL1 (*Variable-valued Logic system 1*, véase [Dietterich y Michalski, 84]), y permite utilizar selectores (igualdad entre un atributo y un valor o conjunto de valores), complejos (conjunciones de selectores) y disyunciones de complejos para construir las reglas de producción.

c) Listas de decisión

Las listas de decisión son otra forma de representación basada en lógica de proposiciones. Es una generalización de los árboles de decisión y de representaciones conjuntivas (CNF) y disyuntivas (DNF). Una lista de decisión es una lista de pares de la forma

$$(d_1, C_1), (d_2, C_2),..., (d_n, C_n)$$

donde cada d_i es una descripción elemental, cada C_i es una clase, y la última descripción C_n es el valor *verdadero*. La clase de un objeto será C_j cuando d_j sea la primera descripción que lo satisface. Por tanto, se puede pensar en una lista de decisión como en una regla de la forma "si d_1 entonces C_1, sino si d_2..., sino si d_n entonces C_n".

Se usan por ejemplo en el sistema CN2 ([Clark y Niblett, 89]) que es una modificación del sistema AQ, que pretende mejorar el aprendizaje a partir de ejemplos con ruido (al evitar la dependencia de ejemplos específicos e incorporar una poda del espacio de búsqueda). Las descripciones elementales de los pares que forman la lista de decisión tienen la misma forma que los *complejos* de AQ.

2.3.1.2 Representaciones basadas en la lógica de predicados de primer orden

Aunque las representaciones basadas en lógica de proposiciones han sido usadas con éxito en muchos sistemas de aprendizaje en ordenadores, tienen algunas importantes limitaciones ([Muggleton, 92]), superadas por la lógica de predicados de primer orden, que restringen su campo de aplicación:

- *Potencia expresiva*: Las representaciones basadas en lógica de proposiciones limitan la forma de los patrones que pueden ser representados, ya que, en general, no pueden expresar relaciones. Así, por ejemplo, no se pueden representar (al menos de un modo sencillo) patrones en los que se cumpla una relación de igualdad entre dos atributos (sólo se permitiría expresar la igualdad entre un atributo y un valor constante).
- *Conocimiento de base*[1]: Es difícil incorporar conocimiento de base en el proceso de aprendizaje. Una forma sencilla de conocimiento de base la constituyen restricciones impuestas a las descripciones generadas por el sistema, aunque ésto puede resultar demasiado restrictivo.
- *Restricciones en el vocabulario*: Las descripciones de los sistemas actuales vienen limitadas por un vocabulario fijo de atributos proposicionales. Podría ser muy útil tener la posibilidad de mejorar la representación mediante la invención de nuevos predicados.

Todas estas limitaciones pueden superarse con una representación del conocimiento más potente: la lógica de predicados de primer orden. Cada vez hay más sistemas de aprendizaje en ordenadores que utilizan de algún modo la lógica de primer orden, surgiendo así un nuevo área de interés llamado *programación lógica inductiva* (en inglés *Inductive Logic Programming* o *ILP*). El objetivo de la programación lógica inductiva es construir un programa lógico (en lógica de predicados de primer orden) que, junto al conocimiento que se tenga del dominio, tenga como consecuencia lógica el conjunto de entrenamiento del sistema.

1. Usaremos el término *conocimiento de base* (en inglés *background knowledge*) para designar todo lo que es conocido de antemano, para realizar el proceso de aprendizaje. Aunque, como señalan [Mira et al., 95] sería más acertado denominarlo *información previa*.

La presente tesis se enmarca dentro del paradigma de la programación lógica inductiva, por ello, en el apartado 2.4, desarrollaremos este punto en mayor detalle, fijándonos en algunos de sus sistemas más representativos.

2.3.1.3 Representaciones estructuradas

Las representaciones estructuradas de conocimiento tienen una gran potencia expresiva (aunque, en teoría, no mayor que la de la lógica de predicados de primer orden) y permiten una fácil interpretación del mismo. Entre las representaciones estructuradas se pueden incluir las *redes semánticas* y los *marcos*. En cualquier caso, el conocimiento expresado mediante una de estas representaciones estructuradas puede ser traducido fácilmente a lógica de predicados de primer orden.

a) Redes semánticas

El término de *red semántica* surge a partir del modelo de memoria semántica de Quillian (1968), con el que pretendía representar el significado de los vocablos del idioma inglés. En una red semántica la representación consta de un conjunto de nodos (conceptos), unidos entre sí por diferentes clases de enlaces asociativos (relaciones). A su vez, las relaciones entre un concepto y su clase, denominadas *relaciones de subtipo* (ej. instancia_de, es_un), a veces se representan en una red separada.

La principal ventaja de las redes semánticas es que toda la información relativa a un objeto concreto se obtiene fácilmente a partir del mismo, siguiendo los arcos que parten de él.

Para aplicar la minería de datos con redes semánticas, se representa cada ejemplo como una red semántica, y las operaciones que se realizan consisten en manipular los grafos, para encontrar los patrones (subgrafos) que cumplen todos los ejemplos de la misma clase.

Es práctica común el denominar *red semántica* a todo formalismo con forma de red usado para representar conocimiento. Sin embargo, es más preciso considerar como tales sólo las que se dedican a la representación del lenguaje natural, y denominar, en general, *redes asociativas* a todas ellas ([Mira et al., 95]).

Una red asociativa es una red (conjunto de nodos unidos entre sí por enlaces) en la que los nodos representan conceptos y los enlaces representan relaciones (de pertenencia, inclusión, causalidad, etc.) entre los conceptos. Dentro de las redes asociativas se incluyen: las *redes semánticas* (destinadas a representar o comprender el lenguaje natural), las *redes de clasificación* (representan conceptos mediante una jerarquía de clases y propiedades) y las *redes causales* (representan relaciones de influencia, generalmente de causalidad, entre variables).

Las redes de clasificación pueden considerarse redes semánticas sólo cuando representan conocimiento taxonómico de conceptos, pero no cuando se utilizan dentro de un sistema experto basado en marcos para definir clases e instancias.

Las redes causales representan un modelo en el que los nodos corresponden a variables y los enlaces a relaciones de influencia, generalmente de causalidad. Los modelos causales se orientan, sobre todo, a problemas de diagnóstico. Las redes bayesianas pueden considerarse como redes causales a las que se ha añadido una distribución de probabilidad sobre sus variables.

b) Marcos

El concepto de marco fue introducido por Minsky (1975) como método de representación del conocimiento y de razonamiento, intentando superar las limitaciones de la lógica a la hora de abordar problemas como la visión artificial, la comprensión del lenguaje o el razonamiento de sentido común ([Mira et al., 95]).

Un marco es una estructura de datos, formada por un nombre y un conjunto de campos (o ranuras, del inglés *slots*), que se rellenan con valores para cada ejemplo concreto. Las ranuras pueden llenarse con valores de atributos o con referencias a otros marcos para expresar las *relaciones de subtipo*, como la relación *es_un*, en cuyo caso se heredan los atributos del marco de nivel superior.

Basándose en el concepto de marco, se desarrolló el lenguaje de programación KRL (*Knowledge Representation Language*), para representar el conocimiento de forma estructurada. La ingeniería del software también heredó de la inteligencia artificial el concepto de marco para construir la orientación a objetos (puede observarse el gran parecido entre los objetos de la programación orientada a objetos y los marcos).

Entre los sistemas de aprendizaje que utilizan marcos para representar el conocimiento adquirido, podemos mencionar el sistema EURISKO ([Lenat., 84]).

El conocimiento expresado mediante marcos puede traducirse fácilmente a lógica de predicados de primer orden, aunque perdiendo la ventaja de ser estructurado.

2.3.1.4 Representaciones basadas en ejemplos

Los sistemas de aprendizaje basado en ejemplos (*Instance-Based Learning algorithms*) representan el conocimiento mediante ejemplos representativos, basándose en "similitudes" entre los datos ([Aha et al., 91]). El aprendizaje consiste en la selección de los ejemplos que mejor representan a los conceptos existentes en la base de datos (se trata de aprendizaje supervisado); estos ejemplos representativos serán los únicos que se almacenen, reduciendo así considerablemente el espacio necesario. El principal problema de estos sistemas es que se necesita una función de "similitud", a veces difícil de definir, para clasificar los nuevos ejemplos según sea su parecido con los ejemplos prototipo.

Los algoritmos de aprendizaje basado en ejemplos surgieron a partir de los clasificadores por vecindad (*nearest-neighbor classifier*), y han adquirido importancia más recientemente con los sistemas de razonamiento basado en casos (*case-based reasoning*), para diagnóstico y resolución de problemas. Además, pueden utilizarse como paso previo a otros sistemas de aprendizaje a partir de ejemplos, para entrenarlos con conjuntos de ejemplos más pequeños y representativos.

2.3.1.5 Redes neuronales

Las redes neuronales ([Lippmann, 87], [Freeman y Skapura, 93], [Hertz et al., 91]), incluidas dentro de los modelos *conexionistas*, son sistemas formados por un conjunto de sencillos elementos de computación llamados *neuronas artificiales*. Estas neuronas están interconectadas a través de unas conexiones con unos pesos asociados, que representan el conocimiento en la red. Cada neurona calcula la suma de sus entradas, ponderadas por los pesos de las conexiones, le resta un valor umbral y le aplica una función no lineal (por ej. una sigmoide); el resultado sirve de entrada a las neuronas de la capa siguiente (en redes como el *perceptrón multicapa*).

Uno de los algoritmos más usado para entrenar redes neuronales es el *back-propagation*, que utiliza un método iterativo para propagar los términos de error (diferencia entre valores obtenidos y valores deseados), necesarios para modificar los pesos de las conexiones interneuronales. El *back-propagation* puede considerarse como un método de regresión no lineal ([Fayyad et al., 96b]), en el que aplica un descenso de gradiente en el espacio de parámetros (pesos), para encontrar mínimos locales en la función de error.

Las redes neuronales han sido utilizadas con éxito en diferentes tipos de problemas:

- Auto-asociación: la red genera una representación interna de los ejemplos aportados, y responde con el más aproximado a su "memoria". Ejemplo: máquina de Boltzman.
- Clasificación de patrones: la red es capaz de clasificar cada entrada en un conjunto predefinido de clases. Ej.: back-propagation.
- Detección de regularidades: la red se adapta a los ejemplos de entrada, tomando de ellos varias características para clasificarlos; en este caso, el conjunto de clases no está definido de antemano, por lo que el aprendizaje es no supervisado. Ej.: red MAXNET, ART1, mapas de Kohonen, red de Oja, etc.

Las tasas de error de las redes neuronales son equivalentes a las de las reglas generadas por los métodos de aprendizaje simbólicos, aunque son algo más robustas cuando los datos son ruidosos. Las principales desventajas para usar redes neuronales en la minería de datos son:

- el aprendizaje es bastante más lento que en un sistema de aprendizaje simbólico ([Shavlik et al., 90]);
- el conocimiento obtenido por las mismas no es representable en forma de reglas inteligibles, sino que lo forma el conjunto de pesos de las conexiones interneuronales;
- además, es difícil incorporar conocimiento de base o interacción del usuario en el proceso de aprendizaje de una red neuronal.

2.3.2 Aprendizaje

2.3.2.1 Enfoques del aprendizaje: conductista y cognoscitivo

Entre la gran variedad de sistemas de aprendizaje desarrollados en ingeniería del conocimiento, se pueden distinguir dos claras tendencias desde un punto de vista psicológico: el enfoque *conductista* y el enfoque *cognoscitivo*. Esto marca diferentes actitudes de los sistemas ante el proceso de aprendizaje, así como su empleo en diferentes aplicaciones y el uso de diferentes lenguajes para representar el conocimiento.

- Sistemas conductistas

  Según la Psicología conductista, el *aprendizaje* es la capacidad de experimentar cambios adaptativos para mejorar el rendimiento. Siguiendo este enfoque, un sistema de aprendizaje será como una caja negra (de la que no interesa su estructura interna) capaz de adecuar su *comportamiento* para que el rendimiento de sus respuestas ante los datos de entrada aumente durante el proceso de aprendizaje.

  Los sistemas de aprendizaje conductistas hacen mayor énfasis en modelos de comportamiento que en la representación interna del conocimiento, que muchas veces es opaca e ininteligible. Los lenguajes de descripción suelen ser diferentes para los objetos y para

el conocimiento: los objetos se describen por vectores de características, mientras que para el conocimiento se emplean parámetros o tablas. En cualquier caso, esta representación del conocimiento hace difícil su traducción a reglas que expliquen de forma racional el comportamiento del sistema.

Las aplicaciones de los sistemas de aprendizaje conductista se extienden por varios campos relacionados con la IA: autómatas de aprendizaje, control adaptativo, reconocimiento de formas y clasificación. En general, presentan buena inmunidad al ruido.

Entre los sistemas conductistas de aprendizaje, hay que destacar los sistemas *conexionistas* y los sistemas *evolucionistas*, que realizan inducción de conocimiento.

- Sistemas cognoscitivos

  Según el enfoque cognoscitivo de la Psicología, el *aprendizaje* consiste en la construcción y modificación de la *representación del conocimiento*. Durante el proceso de aprendizaje se produce un incremento del conocimiento, que no supone un simple cambio cuantitativo, sino también cualitativo; es decir, no hay un mero aumento del volumen de conocimiento almacenado, sino, sobre todo, una *reorganización* del mismo.

  La "calidad" del aprendizaje vendrá dada no sólo por el aumento de precisión del conocimiento almacenado (en una base de conocimiento), sino también por la utilidad del mismo para los objetivos del usuario y por el nivel de abstracción empleado. Por tanto, la representación del conocimiento jugará un papel principal en los sistemas que sigan este enfoque.

  Los lenguajes de descripción usados por estos sistemas suelen coincidir para representar a los objetos y al conocimiento. Están basados, normalmente, en la lógica (de proposiciones o de predicados) o en representaciones estructuradas (como los marcos).

  Las aplicaciones de los sistemas de aprendizaje cognoscitivo dependen del tipo de aprendizaje que realicen, siendo los más importantes los que utilizan deducción (sistemas EBL, basados en explicaciones), analogía (sistemas expertos basados en casos) o inducción (adquisición y formación de conceptos). Los sistemas inductivos en los que el conocimiento se expresa mediante reglas sencillas se suelen denominar *simbólico*, y tienen gran importancia para construir bases de conocimiento en sistemas expertos y para extraer conocimiento de grandes bases de datos.

2.3.2.2 Tipos de aprendizaje

En cualquier proceso de aprendizaje, el aprendiz aplica el conocimiento poseído a la información que le llega, procedente de un maestro o del entorno, para obtener nuevo conocimiento, que es almacenado para poder ser usado posteriormente.

Dependiendo del esfuerzo requerido por el aprendiz (o número de inferencias que necesita sobre la información que tiene disponible) han sido identificadas varias estrategias, aunque, en la práctica, muchos procesos de aprendizaje aplican de forma simultánea varias de ellas. Una clasificación ya "clásica" de los diferentes tipos de aprendizaje, en orden creciente de esfuerzo inferencial por parte del aprendiz, es ([Michalski, 87]):

1. Aprendizaje por implantación directa[1] (*rote learning*): Es un caso extremo, en el que el aprendiz no ha de realizar ningún tipo de inferencia sobre la información suministrada, sino que la acepta directamente. Esta estrategia incluye aprendizaje por programación y por memorización.
2. Aprendizaje por instrucción: El sistema de aprendizaje adquiere el nuevo conocimiento a través de la información proporcionada por un maestro, pero no la copia directamente en memoria, sino que selecciona los datos más relevantes y/o los transforma a una forma de representación más apropiada.
3. Aprendizaje por deducción: Partiendo del conocimiento suministrado y/o poseído, se deduce el nuevo conocimiento, es decir, se transforma el conocimiento existente mediante una función preservadora de la verdad.
4. Aprendizaje por analogía: Se adquiere un nuevo concepto mediante la modificación de la definición ya conocida de un concepto similar. El aprendizaje por analogía puede ser entendido como una combinación de la inducción y la deducción, ya que mediante la inferencia inductiva se determinan características comunes a los dos conceptos comparados, unificando la misma definición para ambos; entonces se aplica la deducción para obtener las características esperadas para el nuevo concepto. Este tipo de aprendizaje es especialmente importante en la resolución de problemas.
5. Aprendizaje por inducción: El sistema de aprendizaje aplica la inducción a los hechos u observaciones suministrados, para obtener nuevo conocimiento. La inferencia inductiva no preserva la verdad del conocimiento, sólo su falsedad; es decir, si partimos de hechos falsos, el conocimiento adquirido por inducción será falso, pero si los hechos son verdaderos, el conocimiento inducido será válido con cierta probabilidad (y no con certeza absoluta, como ocurre con la deducción). Hay dos tipos de aprendizaje inductivo:
   - Aprendizaje con ejemplos: el nuevo conocimiento es inducido mediante la generalización a partir de una serie de ejemplos y contraejemplos. Este método también se conoce como *adquisición de conceptos*.
   - Aprendizaje por observación y descubrimiento: el sistema de aprendizaje analiza una serie de entidades y determina que algunas tienen características comunes, por lo que pueden ser agrupadas formando un concepto. Puesto que los conceptos no son conocidos de antemano, este método se llama también aprendizaje *no supervisado* o *formación de conceptos*.

2.4 Programación Lógica Inductiva

Según [Quinlan y Cameron-Jones, 95], la teoría de lenguajes de primer orden ha sido utilizada desde hace más de 30 años para representar el conocimiento en los sistemas de aprendizaje simbólico. Los sistemas de *aprendizaje basado en explicaciones* (*EBL systems*) siempre han requerido de este tipo de lenguajes, aunque sólo se comenzaron a utilizar en el contexto del aprendizaje inductivo desde 1983. Sin embargo, el aprendizaje empírico de primer orden, incluyendo lo que ahora se llama *programación lógica inductiva* (*Inductive Logic Programming* o *ILP*) no atrajo la atención generalizada hasta los años 90, como puede comprobarse en la figura 2-2, en la que se representa un histograma de referencias bibliográficas sobre programación

1. Parece que Michalski incluye aquí la *implantación directa* para completar la tabla pero, en general, nadie la considera un tipo de aprendizaje.

lógica inductiva, publicadas desde 1930 y recopiladas por [De Raedt y Muggleton, 93] (de un total de 325 títulos).

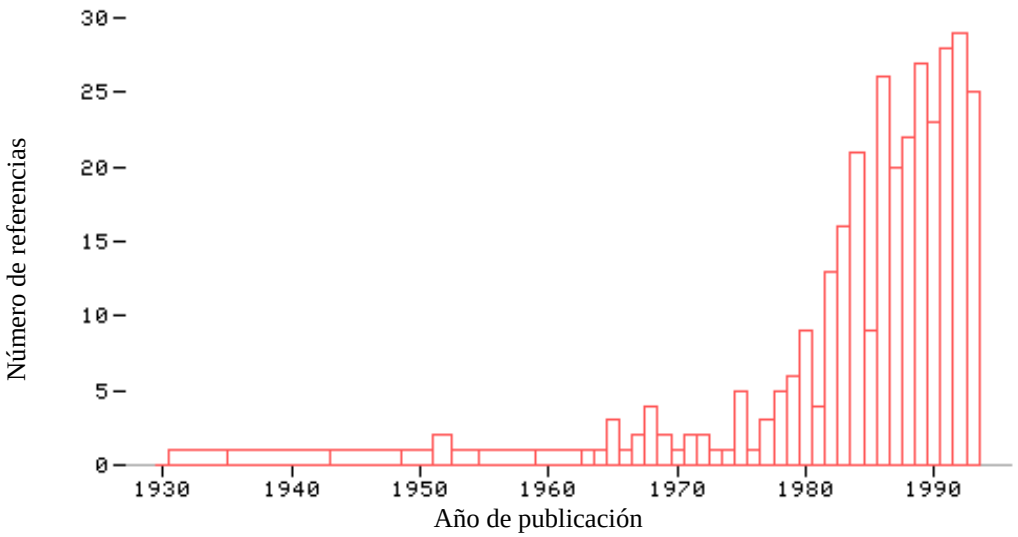

Figura 2-2: Referencias bibliográficas sobre ILP

La programación lógica inductiva se define en [Muggleton, 92] como la intersección entre el aprendizaje inductivo y la programación lógica. Esto es así porque utiliza técnicas de ambos campos:

- Del aprendizaje inductivo en los ordenadores hereda su objetivo: desarrollar herramientas y técnicas para inducir hipótesis a partir de observaciones (ejemplos) y sintetizar nuevo conocimiento a partir de la experiencia.
- De la programación lógica hereda básicamente el formalismo de representación y su orientación a la semántica. La ILP utiliza la programación lógica como mecanismo para representar las hipótesis y las observaciones, superando así dos de las principales limitaciones de las técnicas clásicas de aprendizaje en ordenadores: la rigidez en la representación del conocimiento (lógica de proposiciones, árboles de decisión, etc.) y la dificultad para expresar conocimiento de base (*background knowledge*).

En los sistemas de aprendizaje de orden-0 el conjunto de entrenamiento consta de vectores de valores, cada uno perteneciente a una clase conocida. El conocimiento inducido permite definir clases en función del valor de los atributos, siendo representable con expresiones de la lógica de proposiciones. En ocasiones, se representa en forma de árbol de decisión: ID3, C4.5, etc. y a veces en forma de reglas: PRISM, C4.5rules, etc.

Por el contrario, en los sistemas de aprendizaje de primer orden, el conjunto de entrenamiento lo forman relaciones definidas de forma *extensional*, y el conocimiento de base lo constituyen otras relaciones, definidas *intensionalmente*[1]. El objetivo del aprendizaje es, en estos sistemas, la construcción de un programa lógico que defina de forma intensional una relación objetivo (extensional) del conjunto de entrenamiento. En este tipo de definiciones lógicas se permite la recursión y algunos cuantificadores, muy útiles cuando se trabaja con objetos estructurados, difíciles de describir en un formato objeto-atributo-valor.

El inconveniente de usar representaciones tan expresivas como la lógica de predicados de primer orden es que, aunque las descripciones que se construyen tienden a ser más sencillas que las obtenidas en lógica de proposiciones, el espacio de búsqueda de las mismas es mucho mayor. Esto hace que la búsqueda de la mejor descripción sea una difícil y costosa tarea, que sólo puede realizarse mediante métodos heurísticos de búsqueda.

2.4.1 Clasificación de los sistemas de ILP

Podemos definir diferentes criterios para clasificar los sistemas de ILP:

- Según el número de conceptos que aprenden simultáneamente: pueden aprender un único concepto o múltiples conceptos (predicados) a la vez.
- Según el número de ejemplos que necesitan para el aprendizaje: aprendizaje de un paso (si necesitan todos los ejemplos disponibles) o secuencial (si se les pueden suministrar los ejemplos uno a uno).
- Si necesita un supervisor durante el aprendizaje: interactivo cuando el supervisor valida la corrección del aprendizaje durante el mismo, o no interactivo en caso contrario.

Aunque cada uno de estos criterios de clasificación es independiente del resto, los sistemas de ILP existentes suelen formar solamente dos grupos:

a) Los sistemas que aprenden un solo concepto, en un paso, y de forma no interactiva. Por ejemplo: FOIL, GOLEM, LINUS, etc.

b) Los sistemas que aprenden múltiples predicados, de forma secuencial y ayudados por el usuario de forma interactiva. Por ejemplo: MIS, CLINT, CIGOL, MOBAL, etc.

Algunos autores ([De Raedt et al., 93]) denominan al primer grupo sistemas de ILP **empíricos**, y al segundo sistemas de ILP **interactivos**.

2.4.2 Métodos de ILP

Se puede considerar la programación lógica inductiva como la búsqueda de cláusulas, consistentes con los ejemplos de entrenamiento, en lenguaje de primer orden.

Dentro del espacio de todas las posibles cláusulas, se puede definir una ordenación entre ellas, dada por la relación de generalización (una cláusula es más general que otra cuando cubre un superconjunto de las tuplas cubiertas por la segunda). Según sea el método con el que se recorre el espacio de posibles cláusulas, podemos distinguir varias técnicas ([Dzeroski, 96]):

1. Estamos siguiendo aquí la interpretación dada por [Michalski, 84], según la cual, cabe distinguir entre el *conjunto de hechos* u observaciones de partida y el *conocimiento de base*. El conocimiento de base está formado tanto por las suposiciones y restricciones en las observaciones de partida o en la forma del conocimiento que se puede inducir, como por cualquier conocimiento del problema que sea relevante. De acuerdo con este criterio, el conjunto de hechos lo forman las relaciones explícitas de la base de datos, mientras que las implícitas pertenecen al conocimiento de base.
Sin embargo, otros autores ([Dzeroski, 96]) consideran un *conjunto de ejemplos*, formado por hechos verdaderos y falsos de un predicado p, y un *conocimiento de base*, compuesto por el resto de predicados existentes (distintos de p) que pueden utilizarse para construir la definición de p. Según esta interpretación, el conocimiento de base vendría definido por todas las relaciones (*intensionales* y *extensionales*), con la excepción de la relación objetivo, que constituiría el conjunto de ejemplos.

generalización relativa menos general, resolución inversa, búsqueda de grafos refinados, modelos de reglas, transformación del problema a formato proposicional, etc.

2.4.2.1 Generalización relativa menos general

La generalización menos general (*least general generalization*) de Plotkin ([Plotkin, 69]) se basa en la idea de que si dos cláusulas c_1 y c_2 son ciertas, la generalización más específica común a ambas $lgg(c_1, c_2)$ será también cierta con bastante plausibilidad. Esto permite realizar generalizaciones de forma conservadora. A las técnicas de generalización a partir de los datos se las denomina también técnicas de búsqueda de abajo hacia arriba (*bottom-up*).

La generalización menos general de dos cláusulas (que puedan generalizarse) es el resultado de aplicar la generalización menos general a cada par de literales de ambas, tanto en el antecedente como en el consecuente. Para cada par de literales, esta operación se realiza comparándolos y sustituyendo por variables los atributos que no coincidan. Por ejemplo,

c_1 = enfermo (pedro) :- hijo_de(pedro, juan), fumador(juan)

c_2 = enfermo (alberto) :- hijo_de(alberto, josé), fumador(josé)

$lgg(c_1, c_2)$ = enfermo(X) :- hijo_de(X, Y), fumador(Y).

La generalización relativa menos general (*rlgg*) de dos cláusulas, es la generalización menos general de ambas, relativa a cierto conocimiento de base. La *rlgg* es la técnica en que se basa el sistema GOLEM ([Muggleton y Feng, 92]).

Normalmente, el número de literales de una *rlgg* crece al menos de forma exponencial con el número de ejemplos existentes. Por ello, GOLEM utiliza restricciones que evitan la introducción de literales redundantes, aunque aún así el número de ellos suele ser grande.

Existen sistemas como CHILLIN ([Zelle et al., 94]) que combinan técnicas de búsqueda de abajo hacia arriba (generalización menos general, como hace GOLEM) con técnicas de arriba hacia abajo (especialización por introducción de nuevos literales, como FOIL). Además CHILLIN es capaz de inventar nuevos predicados; cuando después de un proceso de generalización y especialización no se ha llegado a una definición consistente, se prueba la invención de un nuevo predicado, que incluya variables cuyos valores permitan discriminar los ejemplos positivos de los negativos.

2.4.2.2 Resolución inversa

La idea básica de la resolución inversa (*inverse resolution*) consiste en invertir la regla de *resolución*, de la inferencia deductiva, obteniendo así un método de generalización ([Muggleton y Buntime, 88]).

Aplicada en lógica proposicional, la resolución establece que dadas las premisas $p \vee \bar{q}$ y $q \vee r$, se deduce que $p \vee r$. En lógica de predicados de primer orden, la resolución requiere sustituciones de variables por valores de atributos.

La resolución inversa utiliza un operador de generalización basado en invertir la sustitución. Por ejemplo, dados los hechos:

b1 = enfermo(pedro)

b2 = hijo_de(pedro, juan)

b3 = fumador(juan)

tomando b1 y b2, y la sustitución inversa θ_1^{-1} = {pedro/Y} se obtiene:

c1 = *enfermo(Y) :- hijo_de(Y, juan)*

tomando ahora c1 y b3, y la sustitución inversa θ_2^{-1} = {juan/X} se obtiene:

c2 = *enfermo(Y) :- hijo_de(Y, X), fumador(X).*

La generalización por resolución inversa se aplica en sistemas interactivos de ILP como MARVIN ([Sammut y Banerji, 86]) y CIGOL. Dentro del marco de la resolución inversa, cabe destacar la posibilidad de invención de nuevos predicados, realizada en CIGOL ([Muggleton y Buntime, 88]), por ejemplo, para compactar las definiciones y conseguir mayor generalización.

2.4.2.3 Búsqueda en grafos refinados

La principal técnica de especialización en ILP es la búsqueda de arriba a abajo (*top-down search*) en grafos refinados (*refinement graphs*). Los sistemas que utilizan esta técnica, comienzan con la cláusula más general, a la que aplican especializaciones hasta que sólo cubra tuplas positivas. Se usa en FOIL ([Quinlan, 90]) y FOCL ([Pazzani et al., 91]), por ejemplo.

Se denominan *grafos refinados* porque las operaciones de especialización (refinado) que se realizan durante la búsqueda se pueden representar en un grafo acíclico y dirigido (figura 2-3). En este grafo, cada nodo representa una cláusula (el nodo superior es la cláusula más general) y cada arco representa una operación de especialización (sustitución de una variable por una constante, o incremento de un nuevo literal en la cláusula en construcción). La búsqueda dentro de éste grafo suele ser heurística, como en FOIL, debido a que se produce una explosión combinatoria en el conjunto de posibles especializaciones, que hace inabordable la búsqueda exhaustiva.

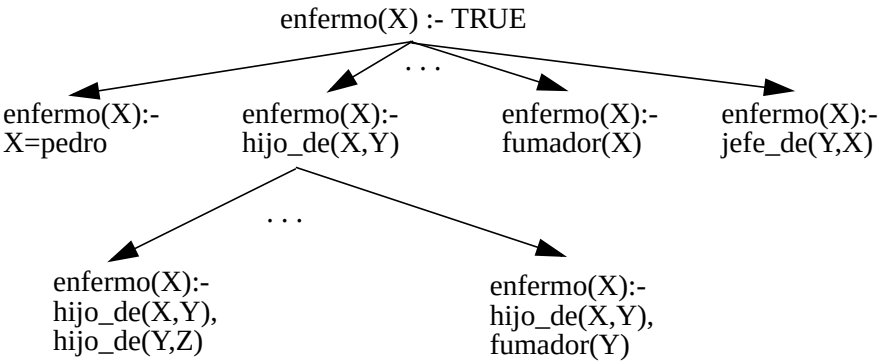

Figura 2-3: Construcción de un grafo refinado

Veamos varios de los sistemas existentes en los que se realiza una búsqueda de grafos refinados:

- INDUCE ([Dietterich y Michalski, 84]) construye descripciones (tanto de objetos como de conceptos) usando predicados, aplicados sobre variables cuantificadas existencialmente, con una lista (que puede estar vacía) de posibles valores para las variables. A estos elementos básicos de las descripciones se les denomina selectores; por ejemplo,

  (tamaño(X) = pequeño, mediano)

  (encima_de(X,Y))

  Podemos distinguir dos tipos de selectores: los que tienen predicados monádicos (de grado uno) y sirven para representar valores de los atributos (por ejemplo, tamaño), y los diádicos (poliádicos en general), que aportan información sobre estructuras (como encima_de).

  Para controlar la explosión combinatoria que se produce en el espacio de búsqueda de descripciones al introducir variables, INDUCE realiza una búsqueda en dos niveles:

  - En primer lugar, explora el subespacio de *sólo estructura*, formado por los predicados monádicos (descriptores de estructuras), construyendo generalizaciones candidatas en sólo estructura, en las que no se asigna valor alguno a las variables utilizadas.
  - A continuación, se explora el subespacio de *atributos*, completando las descripciones de sólo estructura con selectores que asignan valores a las variables.

  Por tanto, INDUCE podría considerarse también como un sistema basado en lógica de proposiciones que permite definir algunas estructuras entre objetos.

- FOIL ([Quinlan, 90]) es un sistema que construye definiciones lógicas formadas por (la disyunción de) varias cláusulas de Horn; para ello parte de datos en formato relacional. FOIL (*First-Order Inductive Learner*) aplica ideas de otros sistemas de aprendizaje en lógica proposicional, extendiéndolas para su uso en lógica de predicados.

  Construye definiciones para una relación de la base de datos, fijada como *relación objetivo* por el usuario, y para ello utiliza el resto de predicados de la base de datos e incluso el predicado objetivo, permitiendo así definiciones recursivas ([Cameron-Jones y Quinlan, 93]).

  En la presente tesis tomaremos el sistema FOIL como punto de partida, sobre el que realizaremos diferentes modificaciones y extensiones para producir resultados con los que se evaluará el trabajo. Por ello, en el apartado 2.5, veremos una descripción más detallada (y reformulada respecto a la de [Quinlan, 90]) de FOIL.

- FOCL ([Pazzani et al., 91]) es una extensión de FOIL que toma ideas de sistemas de *aprendizaje basado en explicaciones* para utilizar conocimiento de base dentro de un proceso inductivo. El conocimiento de base manejado por FOCL lo forman definiciones incorrectas e incompletas, relaciones definidas intensionalmente y *clichés* relacionales.

  Las definiciones incorrectas e incompletas que se le suministran a FOCL son corregidas sin más que recorrer de nuevo el grafo correspondiente y aplicar operaciones de generalización o especialización donde haga falta ([Pazzani y Kibler, 92]).

  Las relaciones intensionales ([Pazzani et al., 91]) son relaciones definidas mediante una cláusula, en vez de por el conjunto de tuplas que la forman. Estas relaciones pueden formar parte de las definiciones construidas, al igual que las relaciones extensionales existentes.

  Por último, los clichés relacionales ([Silverstein y Pazzani, 91]) son esquemas formados por una secuencia de literales que tienden a aparecer juntos en las definiciones.

- Existen variantes de FOIL como mFOIL ([Dzeroski, 91]), adaptado para manejar datos con ruido. La diferencia principal consiste en que mFOIL no construye las cláusulas mediante un heurístico basado en la ganancia de información (o reducción de entropía), sino que selecciona los literales basándose en una estimación del error esperado, a partir del conjunto de entrenamiento. La estimación del error se realiza con probabilidades bayesianas, estimadores de Laplace y m-estimadores.
- Otras variantes, como CHAM ([Kijsirikul et al., 91]), modifican el heurístico de evaluación de literales usado en FOIL, introduciendo un nuevo término que, junto a la medida de la ganancia de FOIL, mida la similitud existente entre dos estados (definidos por las variables usadas en una cláusula). Esta modificación permite construir algunas definiciones sin necesidad de introducir literales determinados ([Quinlan, 91]).
- Algunos sistemas incluyen mecanismos para inventar nuevos predicados durante la construcción del grafo, tratando así de superar las limitaciones del vocabulario (predicados) existente. Esto se hace, por ejemplo, en SIERES ([Wirth y O'Rorke, 92]).

  SIERES, al igual que FOIL, comienza la búsqueda de definiciones a partir de la cláusula más general, a la que va añadiendo literales hasta que es consistente. Sin embargo, en SIERES se imponen una serie de restricciones a los literales de las cláusulas, concretamente en los posibles argumentos que pueden utilizar dichos literales (existen unos grafos de dependencias entre los argumentos, que deben cumplir todos los literales). Estas restricciones a veces resultan demasiado restrictivas e impiden encontrar definiciones para algunos predicados; pero, por otro lado, facilitan la invención de nuevos predicados.

  La necesidad de inventar un nuevo predicado se produce cuando, durante la construcción de una cláusula, no es posible encontrar ningún literal que complete algún grafo de dependencias de argumentos. Entonces se inventa un nuevo predicado, cuyos argumentos vienen impuestos por el propio grafo de dependencias. A continuación se aplica la abducción para generar los ejemplos del nuevo predicado y, finalmente, se aplica de forma recursiva el algoritmo para inducir la definición del nuevo predicado.
- HYDRA ([Ali y Pazzani, 93]) es un sistema que extiende la funcionalidad de FOIL, para poder definir simultáneamente varios conceptos (clases) existente en el conjunto de entrenamiento, y no sólo la relación objetivo.

  La forma de construir cláusulas en HYDRA es similar a la que se sigue en FOIL (aunque los literales candidatos se evalúan con una función algo diferente). La principal diferencia está en que en HYDRA se permite la existencia simultánea de varias cláusulas (una para cada clase existente) durante el proceso de aprendizaje. Una vez construidas todas las definiciones, se asigna a cada cláusula una medida de su *suficiencia*, calculada como la proporción entre ejemplos positivos y negativos cubiertos. Finalmente, las cláusulas construidas competirán para clasificar los ejemplos, ganando aquélla con mayor fiabilidad (suficiencia) que cubra cada ejemplo dado; la cláusula ganadora decidirá la clase de cada nuevo ejemplo.

  El objetivo de este método es construir definiciones más compactas y fiables cuando los datos son ruidosos, pues parece demostrado que la asignación de pesos y la competencia entre cláusulas reduce la sensibilidad al ruido. Existen variantes de este sistema, como HYDRA-MM ([Ali et al., 94]), para construir múltiples definiciones de un mismo concepto.

- El aprendizaje de definiciones para *múltiples predicados* puede considerarse similar al aprendizaje de un único predicado, aunque, en ocasiones, implica algunas dificultades adicionales. Por ejemplo, cuando dos definiciones son mutuamente recursivas (es decir, la primera definición es dependiente de la segunda y, a su vez, la segunda de la primera), entonces nos encontramos con que no son aplicables en la práctica. Por ejemplo,

  abuelo_de(X,Y) :- padre_de(X,Z),padre_de(Z,X).

  padre_de(X,Y):-padre_de(Z,X),abuelo_de(Z,Y),¬tio_de(X,Y).

  Aunque ambas definiciones son correctas, no son aplicables para deducir nuevo conocimiento, pues se produciría una recursión infinita al ejecutarlas.

  La solución seguida por el sistema MPL (*Multiple Predicate Learner*) en [De Raedt et al., 93] considera todas las posibles secuencias de aprendizaje para los diferentes predicados, hasta encontrar un conjunto de definiciones que sea globalmente consistente. Cada una de las definiciones individuales debe ser también localmente consistente y, para ello, se realiza una búsqueda similar a la de FOIL y mFOIL.

2.4.2.4 Patrones de reglas

Se denominan patrones de reglas a un tipo especial de reglas en las que los predicados son variables. Se usan en el sistema MOBAL ([Sommer et al., 93]).

La búsqueda de reglas se realiza a partir de los patrones de reglas existentes, bien proporcionados por el usuario o bien derivados de otras reglas aprendidas previamente por el sistema. Para cada patrón de regla se prueban todas las posibles combinaciones de predicados existentes y, cada una de las reglas así obtenidas, se evalúa con los datos de entrenamiento. Por ejemplo, con un patrón de reglas de la forma

P(X) :- R(X,Y), Q(Y)

se podría obtener, por ejemplo, la regla

enfermo(X) :- hijo_de(X,Y), fumador(Y)

2.4.2.5 Transformación de problemas ILP a formato proposicional

El sistema LINUS ([Dzeroski y Lavrac, 91]) se basa en la idea de convertir problemas de formato relacional a formato proposicional, utilizar sistemas de aprendizaje basados en lógica de objeto-atributo-valor y, finalmente, volver a convertir el conocimiento obtenido a formato relacional.

Este proceso de transformación sólo puede realizarse para cierta clase de problemas: los representables mediante cláusulas de Horn libres de funciones, tipadas (cada variable tiene asociado un dominio finito), restringidas (no se permiten variables nuevas en el cuerpo de las cláusulas) y no recursivas (el predicado del consecuente no puede aparecer en ningún antecedente). Es decir, se reduce al lenguaje de las bases de datos deductivas jerárquicas, con la restricción adicional de no introducir variables nuevas en el cuerpo de las cláusulas.

La ventaja de LINUS consiste en utilizar un lenguaje basado en la lógica de predicados (aunque con ciertas restricciones) y, a la vez, favorecerse de las técnicas desarrolladas para inducción de conocimiento en formato proposicional, por ejemplo de algoritmos como CN2 para manejar datos con ruido.

La forma de pasar de formato relacional a formato proposicional consiste en crear una nueva relación, extendiendo cada tupla de la relación objetivo con un atributo adicional para cada

posible literal (combinando cada predicado de la base de datos con cada variable del consecuente). Por ejemplo, si se quiere buscar una definición para *madre_de(X,Y)*, conociendo otros predicados como *progenitor(X,Y)*, *varon(X)* y *mujer(X)*, se construiría una relación de la forma:

clase	**X**	**Y**	**prog(X,X)**	**prog(X,Y)**	**...**	**mujer(X)**	**mujer(Y)**
+	sofía	elena	falso	verdad		verdad	verdad
+	sofia	felipe	falso	verdad		verdad	falso
-	juan	felipe	falso	verdad		falso	falso
...							

Si el sistema de aprendizaje proposicional induce, por ejemplo, la regla:

[clase = +] si [progenitor(X,Y) = verdad] $\wedge$ *[mujer(X) = verdad]*

entonces, podemos pasar de nuevo al formato relacional del siguiente modo:

madre_de(X,Y) :- progenitor(X,Y), mujer(X)

2.4.2.6 Descubrimiento de cláusulas

Normalmente el objetivo de los sistemas de ILP es la inducción de reglas que definan ciertos predicados. Estas reglas deberían ser lo suficientemente precisas como para poder reemplazar a los ejemplos de la base de datos y que éstos fueran deducidos a partir de las mismas.

Pero puede considerarse un enfoque diferente, conocido como *ILP no monótono*, que consiste en la búsqueda de restricciones dentro de la base de datos. Estas restricciones pueden tener la forma de una cláusula en lógica de predicados de primer orden, por lo que se este enfoque se denomina *descubrimiento de cláusulas* (*clausal discovery*). Este tipo de inducción se realiza en sistemas como CLAUDIEN ([De Raedt y Bruynooghe, 93]). Algunos ejemplos de reglas que pueden ser inducidas por CLAUDIEN son las siguientes:

padre_de(X,Y) $\vee$ *madre_de(X,Y) :- progenitor_de(X,Y)*

progenitor_de(X,Y) :- padre_de(X,Y)

false :- padre_de(X,Y), madre_de(X,Y)

La última de ellas se interpreta como que una persona no puede ser a la vez padre y madre de otra.

2.5 El sistema FOIL

2.5.1 Idea básica

El objetivo de FOIL (*First-Order Inductive Learner*) es obtener una definición lógica de una relación P en función de otras relaciones Q_i y de ella misma, permitiendo así definiciones

recursivas. Tanto la relación P, que es fijada como *relación objetivo* de antemano, como las demás relaciones Q_i están definidas inicialmente de forma extensional, es decir, por los conjuntos de tuplas para los que se satisfacen los correspondientes predicados.

El resultado obtenido por FOIL es una regla o definición lógica (intensional) de la relación objetivo P, mediante un conjunto de cláusulas de Horn con cabeza, para verificar cuándo se satisface el predicado p. Una definición puede estar formada por una sola cláusula de Horn o por la disyunción de varias, siendo la forma general de estas cláusulas la siguiente:

$$L_1 \wedge L_2 \wedge ... \wedge L_n \rightarrow p(V_1, ..., V_k)$$

aunque normalmente usaremos la siguiente notación equivalente para representarlas:

$$p(V_1, ..., V_k) :- L_1, L_2, ..., L_n$$

donde

- la cabeza (consecuente) de cada cláusula es siempre el predicado objetivo p, de grado k, con todos sus términos $(V_1, ..., V_k)$ variables;
- el cuerpo es la conjunción de una serie de literales $(L_1, L_2, ..., L_n)$.

Los literales del cuerpo (antecedentes) de las cláusulas pueden ser:

a) predicados correspondientes a alguna de las relaciones Q_i definidas inicialmente (incluida la relación objetivo P), construidos sobre términos variables;

b) o bien, el predicado igualdad ("=") o el predicado "mayor que" (">") aplicados sobre un par de variables existentes, o comparando una variable con un valor constante.

Los literales del cuerpo pueden llevar el signo "¬" de la negación lógica.

2.5.2 Algunas definiciones

Veremos las definiciones de algunos conceptos generales, necesarios para explicar de forma precisa el algoritmo FOIL y sus diferentes heurísticos.

2.5.2.1 Signo de una tupla

Decimos que el signo de una tupla $t = <a_1,..., a_k>$ es "+", o simplemente que t es *positiva* o ejemplo de p (de grado k), cuando su proyección sobre los argumentos del predicado objetivo p está incluida en la relación P asociada. En caso contrario, decimos que el signo de t es "–", o que es *negativa* o contraejemplo de p:

$$t \in T^+ \Leftrightarrow \models_t (p(V_1,..., V_k)) \qquad \text{[EC. 2.1]}$$

$$t \in T^- \Leftrightarrow t \notin T^+ \text{ (suposición de mundo cerrado)}^1 \qquad \text{[EC. 2.2]}$$

El signo de una tupla t' que sea extensión de otra tupla t (apartado 2.5.2.2) será el mismo que el de esta última.

1. La suposición de mundo cerrado o *closed world assumption* establece que todas las tuplas pertenecientes a P se declaran de forma explícita, por tanto, las no definidas se supondrán no pertenecientes a P. En FOIL es posible también declarar explícitamente las tuplas que no pertenecen a la relación P.

El conjunto de entrenamiento T para inducir una definición lo construimos con los conjuntos de ejemplos y contraejemplos del predicado objetivo:

$$T = T^+ \cup T^- \qquad \text{[EC. 2.3]}$$

2.5.2.2 Extensión de una tupla

Decimos que una tupla t' = <a_1,..., a_m, b_1,..., b_n>, de tamaño m+n, es una *extensión* de otra tupla t = <a_1,..., a_m>, de tamaño m, cuando los m primeros valores de t' coinciden con los de t.

Como veremos más adelante, cuando se añaden literales con variables nuevas a una cláusula se obtienen conjuntos locales de entrenamiento cuyas tuplas son extensiones de las del conjunto inicial de entrenamiento. Las tuplas que son extensión de otra t conservan el signo de t.

2.5.2.3 Satisfacción

Decimos que una tupla t *satisface un literal* L = q(V_1,..., V_n), o que L se satisface para t, y lo expresamos como $\models_t$(L), cuando al asignar a las variables (V_1,..., V_n) de L los valores de t obtenemos una nueva tupla t' que pertenece a la relación Q:

$$\models_t (L) \quad \Leftrightarrow \quad <t(V_1), t(V_2),..., t(V_n)> \in Q \qquad \text{[EC. 2.4]}$$

Decimos que una tupla t *satisface una cláusula* C = [L_0 :- L_1,..., L_n], o que C se satisface para t, y lo expresamos como $\models_t$(C), siempre que no ocurra que t satisface a todos los antecedentes ($L_1 \wedge ... \wedge L_n$) pero no al consecuente (L_0):

$$\models_t (C) \quad \Leftrightarrow \quad \neg(\models_t(L_1 \wedge ... \wedge L_n) \wedge \neg \models_t(L_0)) \qquad \text{[EC. 2.5]}$$

aplicando las leyes de DeMorgan, podemos reescribir esta definición como: C se satisface para t cuando t es positiva o cuando C no cubre a t (ver [EC. 2.7])

$$\models_t (C) \quad \Leftrightarrow \quad \neg \models_t(L_1 \wedge ... \wedge L_n) \vee \models_t(L_0) \Leftrightarrow \qquad \text{[EC. 2.6]}$$
$$\Leftrightarrow \quad (t \notin T_C(C)) \vee (t \in T^+)$$

2.5.2.4 Conjunto cubierto

Decimos que una cláusula de Horn C = [L_0 :- $L_1 \wedge ... \wedge L_n$] *cubre a una tupla* t cuando ésta satisface a todos sus antecedentes:

$$C \text{ cubre a } t \quad \Leftrightarrow \quad \models_t(L_1 \wedge L_2 \wedge ... \wedge L_n) \qquad \text{[EC. 2.7]}$$

El *conjunto cubierto por una cláusula* C lo forman todas las tuplas del conjunto de entrenamiento T cubiertas por la misma:

$$T_C(C) = \{t \in T \mid \models_t(L_1 \wedge L_2 \wedge ... \wedge L_n)\} \qquad \text{[EC. 2.8]}$$

El *conjunto cubierto por una definición* D ≡ [$C_1 \vee ... \vee C_m$] lo forma la unión de los conjuntos cubiertos por cada una de sus cláusulas:

$$T_C(D) = T_C(C_1) \cup ... \cup T_C(C_m) \qquad \text{[EC. 2.9]}$$

2.5.2.5 Relación residual

La *relación residual* asociada a una cláusula C la forman todas las tuplas positivas no cubiertas por C:

$$T_R(C) = T^+ - T_C(C) = \{t \in T^+ \mid \neg \models_t (L_1 \wedge L_2 \wedge ... \wedge L_n)\} \quad \text{[EC. 2.10]}$$

La *relación residual* asociada a una definición $D \equiv [C_1 \vee ... \vee C_m]$ la forman las tuplas positivas no cubiertas por ninguna de sus cláusulas:

$$T_R(D) = T_R(C_1) \cap ... \cap T_R(C_m) \quad \text{[EC. 2.11]}$$

2.5.2.6 Consistencia

Decimos que una cláusula $C = [L_0 :- L_1 \wedge ... \wedge L_n]$ es *consistente* sobre el conjunto T cuando no cubre a ninguna tupla negativa de T:

$$\text{C es consistente sobre T} \Leftrightarrow (\forall t \in T^-)\ (\neg \models_t (L_1 \wedge L_2 \wedge ... \wedge L_n)) \quad \text{[EC. 2.12]}$$
$$\Leftrightarrow (\forall t \in T^-)\ (t \notin T_C(C))$$

Esta misma condición se puede reescribir como:

$$\text{C es consistente sobre T} \Leftrightarrow \left| T_C(C) \cap T^- \right| = 0 \quad \text{[EC. 2.13]}$$

o bien, considerando que $T_C(C) \subseteq T$, también se puede reescribir como:

$$\text{C es consistente sobre T} \Leftrightarrow \left| T_C(C) \cap T^+ \right| = \left| T_C(C) \right| \quad \text{[EC. 2.14]}$$

A partir de esta última expresión se puede generalizar hacia una condición de k-consistencia, menos estricta, que cumplen todas las cláusulas consistentes y aquéllas inconsistentes cuya precisión supera cierto umbral k definido en [0, 1] (véase apartado 2.5.4.5):

$$\text{C es k-consistente sobre T} \Leftrightarrow \left| T_C(C) \cap T^+ \right| \geq k \cdot \left| T_C(C) \right| \quad \text{[EC. 2.15]}$$

en el caso de k = 1, esta condición de k-consistencia se reduce a la de consistencia ([EC. 2.14]).

La consistencia de una definición D se define de modo similar a la de las cláusulas:

$$\text{D es consistente} \Leftrightarrow (\forall t \in T^-)\ (t \notin T_C(D)) \quad \text{[EC. 2.16]}$$

2.5.2.7 Completitud

Decimos que una cláusula $C = [L_0 :- L_1 \wedge ... \wedge L_n]$ es *completa* sobre el conjunto T cuando cubre a todas las tuplas positivas de T:

$$\text{C es completa sobre T} \Leftrightarrow (\forall t \in T^+)\ (\models_t (L_1 \wedge L_2 \wedge ... \wedge L_n)) \quad \text{[EC. 2.17]}$$

Esta condición también se puede expresar en función del conjunto cubierto:

$$\text{C es completa sobre T} \Leftrightarrow T^+ \subseteq T_C(C) \quad \text{[EC. 2.18]}$$

o de la relación residual asociada a C:

$$\text{C es completa sobre T} \quad \Leftrightarrow T_R(C) = \varnothing \qquad \text{[EC. 2.19]}$$

Del mismo modo, una definición D será completa cuando cubra todas las tuplas positivas:

$$\begin{aligned} \text{D es completa} \quad & \Leftrightarrow (\forall t \in T^+)\ (t \in T_C(D)) \\ & \Leftrightarrow T_C(D) = T^+ \\ & \Leftrightarrow T_R(D) = \varnothing \end{aligned} \qquad \text{[EC. 2.20]}$$

2.5.2.8 Longitud de descripción extensional

Denominamos *longitud de descripción extensional*[1] (LDE) al número de bits necesarios para codificar un conjunto de tuplas o ejemplos. Cuando este conjunto de tuplas corresponda a una relación, hablaremos de longitud de descripción extensional de la misma; cuando se refiera al conjunto cubierto por una cláusula (o una definición), hablaremos de longitud de descripción extensional de la cláusula (o definición).

Para una relación finita Q, considerada como un conjunto de tuplas, se puede definir la longitud de descripción extensional como el número de bits necesarios para enumerar sus p tuplas, siendo N la cardinalidad del conjunto universal asociado:

$$LDE(Q) = \log_2(N) + \log_2\left(\binom{N}{p}\right) \text{ bits} \qquad \text{[EC. 2.21]}$$

La longitud de descripción extensional de una cláusula C vendrá dada por el número de bits necesarios para enumerar las $|T_C^+(C)|$ tuplas positivas del conjunto $T_C(C)$ cubierto por ella:

$$LDE(C) = \log_2(N) + \log_2\left(\binom{N}{|T_C^+(C)|}\right) \text{ bits} \qquad \text{[EC. 2.22]}$$

siendo

$N = \|T_1\|$	el tamaño del conjunto de entrenamiento
$\|T_C^+(C)\|$	el número de tuplas positivas cubiertas.

1. El significado intensional se refiere a la *definición* de un concepto, mientras que el extensional se refiere a los elementos a los que se les puede aplicar el concepto *dentro de un universo determinado* ([Mira et al., 95]). Por ejemplo, para “planetas solares”:
- significado intensional: planetas que giran alrededor del Sol;
- significado extensional: {Mercurio, Venus, Tierra, Marte, Júpiter, Saturno, Urano, Neptuno, Plutón}

El significado intensional no varía (salvo que se redefina el concepto). Sin embargo el alcance extensional puede variar en el tiempo: el número de planetas puede aumentar o disminuir por un fenómeno cósmico o porque aumente nuestro conocimiento de la realidad.

Por extensión, podemos definir la longitud de descripción extensional de una definición D (disyunción de cláusulas) como:

$$LDE(D) = \log_2(N) + \log_2\left(\begin{pmatrix} N \\ |T_C^+(D)| \end{pmatrix}\right) \text{ bits} \qquad \text{[EC. 2.23]}$$

siendo

$N = |T|$ el tamaño del conjunto de entrenamiento

$|T_C^+(D)|$ el número de tuplas positivas cubiertas.

2.5.2.9 Longitud de descripción intensional

Denominaremos *longitud de descripción intensional* (LDI) al número de bits necesarios para codificar una descripción intensional. En el caso de un literal, será el número de bits necesarios para describir al mismo dentro del espacio de literales candidatos. En el caso de cláusulas y definiciones, se calculará a partir de los literales que las constituyen.

Para un literal L_i la longitud de descripción intensional será:

$$LDI(L_i) = 1 + \log_2(NR) + \log_2(NA) \text{ bits} \qquad \text{[EC. 2.24]}$$

donde el primer sumando corresponde al signo (negado o no), el segundo el predicado (suponiendo que hay NR relaciones en la base de datos) y el último los argumentos del literal (suponiendo NA posibles argumentos).

La longitud de descripción intensional de una cláusula C^n (con n literales) viene dada por la suma de las longitudes de descripción intensionales de sus literales, reducida en factorial de n (pues el orden de los literales no afecta al significado de la cláusula):

$$LDI(C^n) = \sum_{i=1}^{n} LDI(L_i) - \log_2(n!) \text{ bits} \qquad \text{[EC. 2.25]}$$

Por último, para una definición D compuesta por n cláusulas C_i, se puede definir una longitud de descripción intensional a partir de la LDI de sus cláusulas:

$$LDI(D) = \sum_{i=1}^{n} LDI(C_i) - \log_2(n!) \text{ bits} \qquad \text{[EC. 2.26]}$$

2.5.3 Algoritmo

El algoritmo de FOIL está formado por dos bucles:

- uno externo, en el que se construyen definiciones completas a partir de cláusulas consistentes;

- uno interno, en el que se construyen cláusulas consistentes.

En cada iteración del bucle interno se explora el espacio de todos los literales candidatos, y se utiliza un heurístico (*Ganancia*) para evaluarlos y seleccionar el mejor.

2.5.3.1 Bucle externo: construcción de definiciones completas

En el bucle externo de FOIL se construye una definición completa mediante la disyunción de cláusulas consistentes; es decir, se cubre el conjunto T^+ con cláusulas consistentes. En cada iteración se añade una nueva cláusula a la definición en construcción y se modifica el conjunto de entrenamiento, eliminando del mismo las tuplas cubiertas por la nueva cláusula. Esta idea es, esencialmente, la misma que se utiliza en otros algoritmos bien conocidos en el campo del aprendizaje, como el sistema AQ ([Michalski, 87]).

```
Definición& FOIL (T+, T−, p, q1, q2,...)
{
    Definicion D0 = FALSE ;
    Conjunto TC0 = Ø;
    Conjunto TR0 = T+;
    i = 1;
    repetir
        Clausula Ci = construyeC (TR i - 1, T−, p, q1, q2,...);
        Di = Di-1 ∨ Ci;                    /* añade cláusula consistente */
        TCi = TCi - 1 ∪ TC(Ci);            /* actualiza TC */
        TRi = T+ - TCi;                    /* actualiza TR */
        i ++;
    hasta que ((TRi-1 == Ø) || poda(Di-1, TCi-1));    /* definición completa o poda */
    return Di-1;
}.
```

El algoritmo finaliza cuando se cumple la condición de *completitud*, concluyendo con una definición completa. En algunos casos, sin embargo, se permite la construcción de definiciones incompletas (cuando se cumple la condición de poda por mínima longitud de descripción, según se indica en el apartado 2.5.4.5).

2.5.3.2 Bucle interno: construcción de cláusulas consistentes

En el bucle interno de FOIL (función *construyeC* del pseudocódigo del bucle externo) se construyen cláusulas consistentes, cuya cabeza es el predicado objetivo aplicado sobre variables, y cuyo cuerpo está formado por la conjunción de una serie de literales. Estos literales son básicamente predicados de la base de datos, o predicados predefinidos (predicados "=" y ">"), debiendo cumplir todos ellos una serie de restricciones (apartado 2.5.4.2).

En cada iteración del bucle interno se añade un nuevo antecedente a la cláusula y se modifica el conjunto de entrenamiento, seleccionando las tuplas que satisfacen el nuevo literal.

El pseudocódigo para este bucle interno es el siguiente:

```
Clausula& construyeC (TR, T−, p, q1, q2,...)
{
```

```
Clausula C^0 = p; /* cláusula inicial = "p :- TRUE" */
Conjunto T_1^+ = T_R;
Conjunto T_1^- = T^-;
i = 1;
repetir
    Literal L_i = buscaAntecedente (T_i, p, q_1, q_2,...) ;
    C^i = C^(i-1) ∨ (¬L_i);          /* añade antecedente[1] L_i */
    T_(i+1) = actualizarT (T_i, L_i);   /* tuplas que satisfacen L_i */
    i ++;
hasta que ((T_i^- == Ø) || poda(C^(i-1), T_1));   /* cláusula consistente o poda */
return C^(i-1);
};
```

La función *actualizarT* crea un nuevo conjunto local de entrenamiento T_{i+1}, seleccionando las tuplas de T_i que satisfacen L_i. Cuando L_i introduce variables nuevas en la cláusula C^i, las tuplas de T_i se expanden en T_{i+1}, aumentando de tamaño, de modo que éste coincide con el número total de variables de C^i. Si los términos nuevos de las tuplas de T_{i+1} pueden tomar varios valores, entonces una tupla de T_i puede dar lugar a más de una en T_{i+1}; es decir,

si $L_i = q(X_1,..., X_m, Y_1,..., Y_n)$

siendo $X_1,..., X_u$ variables previamente usadas en la cláusula $(m \leq u)$

$Y_1,..., Y_n$ variables nuevas introducidas por L_i

entonces $T_i = \{t = <c_1,..., c_u> \mid \models_t(L_1 \wedge ... \wedge L_{i-1})\} \Rightarrow$

$\Rightarrow T_{i+1} = \{t = <c_1,..., c_u, d_1,..., d_n> \mid \models_t(L_1 \wedge ... \wedge L_i)\}$

Por tanto, el conjunto intermedio T_i se puede definir como el conjunto de tuplas expandidas cubiertas por C^{i-1}, y sirve como conjunto de entrenamiento para seleccionar el siguiente L_i y construir la siguiente cláusula C^i.

La construcción de una cláusula finaliza cuando ésta es consistente, es decir, cuando sólo cubre tuplas positivas. Sin embargo, existe un heurístico de poda por mínima longitud de descripción (apartado 2.5.4.5) que, en ocasiones, permite la existencia de cláusulas inconsistentes dentro de una definición.

2.5.3.3 Evaluación de literales

El objetivo de la función *buscaAntecedente* del pseudocódigo del bucle interno de FOIL es explorar el espacio de los literales candidatos para elegir el mejor de ellos.

Este espacio de búsqueda se recorre de forma casi exhaustiva, intentando simplificar el proceso cuando sea posible mediante la aplicación de una serie de heurísticos: restricciones en la forma de los literales candidatos (apartado 2.5.4.2) y poda *alpha-beta* (apartado 2.5.4.3).

1. La cláusula $C^i = [p:-L_1,..., L_i]$ expresada en forma clausulada es:
$(\neg L_1) \vee ... \vee (\neg L_i) \vee p$
por tanto es equivalente a:
$C^{i-1} \vee (\neg L_i)$

Para medir la bondad de cada literal dentro del espacio de búsqueda, Quinlan utiliza en FOIL un heurístico (*Ganancia*) basado en una medida de la información; de modo similar se comporta otro sistema suyo muy conocido, ID3 ([Quinlan, 86]). El aspecto de esta función de evaluación lo veremos en el apartado 2.5.4.1.

2.5.4 Heurísticos usados en FOIL

Entre los heurísticos que se utilizan en FOIL, el más importante es la función *Ganancia*, que evalúa los literales candidatos para decidir cuál es el mejor (apartado 2.5.3.3). Además existen otros heurísticos:

- Heurísticos para guiar la búsqueda de antecedentes: se usan en la función *buscaAntecedente* del bucle interno. Los hay para acortar la exploración de literales candidatos y para extender la búsqueda a otros literales con variables nuevas:
  - para simplificar la búsqueda se usan dos heurísticos: restricción en la forma de los candidatos y poda *alpha-beta*;
  - para extender el espacio de búsqueda se utilizan los literales determinados.
- Heurísticos para limitar la complejidad de las descripciones: permiten la construcción de definiciones inconsistentes y/o incompletas; se usa tanto en el bucle externo (apartado 2.5.3.1) como en el bucle interno (apartado 2.5.3.2).

2.5.4.1 Ganancia de un literal

La función *Ganancia* está basada en una medida de la información de los conjuntos de entrenamiento T_i, empleados durante la construcción de las cláusulas en el sistema FOIL.

El conjunto de entrenamiento T_i puede ser entendido como un sistema discreto de información ([Cendrowska, 88]), es decir, contiene un número finito de mensajes (tuplas con signo positivo o negativo) que aportan información sobre un evento (pertenencia o no a la relación objetivo P). Se define la cantidad de **información** aportada por un mensaje acerca de un evento como

$$\text{Info(sistema)} = \log_2(\text{Pro}^{\text{después}} / \text{Pro}^{\text{antes}}) \text{ bits} \qquad \text{[EC. 2.27]}$$

siendo $\text{Pro}^{\text{antes}}$ y $\text{Pro}^{\text{después}}$ las probabilidades de ocurrir un evento antes y después de recibir un mensaje en el sistema.

Dado que las reglas que construimos pretenden caracterizar la relación P, nos interesarán los mensajes que informen precisamente sobre el cumplimiento de P, es decir, las tuplas positivas. En ese caso, las probabilidades de que una tupla satisfaga la relación P serán:

$\text{Pro}^{\text{después}} = 1$ probabilidad de que tupla + satisfaga p

$\text{Pro}^{\text{antes}} = N_i^+/(N_i^+ + N_i^-)$ probabilidad de que una tupla satisfaga p

Por tanto, la información aportada por cada una de las tuplas positivas en el conjunto T_i será[1]:

1. Suponemos que el conjunto de entrenamiento local T_i es el que sirve para evaluar y seleccionar el literal i-ésimo L_i de la cláusula en construcción. Las tuplas de T_i que satisfacen L_i son seleccionadas para formar el nuevo conjunto T_{i+1}, tal como se describe en el apartado 2.5.3.2

$$\text{Info}(T_i) = \log_2\left(\frac{1}{N_i^+ / (N_i^+ + N_i^-)}\right) \text{ bits} = -\log_2\left(\frac{N_i^+}{N_i^+ + N_i^-}\right) \text{ bits} \qquad \text{[EC. 2.28]}$$

donde utilizamos la siguiente notación:

$N_i^+ = |T_i^+|$ número de tuplas + de T_i

$N_i^- = |T_i^-|$ número de tuplas – de T_i

La *Ganancia* de un literal L_i se define ([Quinlan, 90]) como:

$$\text{Ganancia}(L_i) = N_i^{++} \cdot (\text{Info}(T_i) - \text{Info}(T_{i+1})) \qquad \text{[EC. 2.29]}$$

siendo

$N_i^{++} \leq N_i^+$ el número de tuplas + de T_i que satisfacen a L_i

La función *Ganancia* es la medida básica usada por FOIL para elegir el siguiente literal L_i, pero, además, asigna un pequeño crédito adicional a los literales no negados que introducen nuevas variables. El motivo de esto es que si ningún literal introduce una ganancia importante, puede ser interesante introducir nuevas variables, ampliando el espacio de búsqueda de literales, y probar de nuevo. De este modo, se permite la formación de cláusulas con literales cuya ganancia sea próxima a cero, que son frecuentes en muchas definiciones correctas; por ejemplo, cuando todos los objetos tienen un valor para cierta relación Q y el literal q(X,Y) define el valor Y para el objeto X, cada tupla de T_i dará lugar a una tupla en T_{i+1} y por ello la ganancia será siempre cero.

Desgraciadamente, puede haber muchos literales que introducen variables nuevas y la elección de uno de ellos es, en estos casos, arbitraria.

2.5.4.2 Restricciones en la forma de los literales candidatos

Se imponen una serie de restricciones en los literales que componen el espacio de búsqueda durante la construcción de una cláusula C^i:

1. Los literales candidatos tomarán una de estas formas posibles:
   - $q(V_1,..., V_m)$, siendo q un predicado del que se conoce su definición extensional (y puede coincidir con el predicado objetivo); las variables V_i pueden haber sido usadas previamente en la cláusula o ser nuevas;
   - $U_i = U_j$, igualdad entre dos variables usadas en la cláusula;
   - $U_i > U_j$, relación de orden entre dos variables;
   - $U_i = c$, igualdad entre una variable y un valor constante;
   - $U_i > c$, comparación entre una variable usada y una constante.

   Cualquiera de estos posibles literales puede llevar además el signo "¬" de la negación lógica.
2. Además, todos los literales candidatos deben tener al menos una variable usada en la cláusula en construcción C^{i-1}. Esto se impone para que cada nuevo literal tenga conexión con los literales anteriormente seleccionados en la cláusula actual; aunque podrían encontrarse, además, otras justificaciones referentes a la eficiencia de cálculo de las cláusulas de Horn, dependiente del orden de sus antecedentes.

3. Si el predicado del literal L_i coincide con el predicado p de la cabeza (predicado objetivo), se restringen los posibles argumentos, para evitar definiciones recursivas inútiles. Esto se hace imponiendo una *ordenación parcial reflexiva*, para evitar que llamadas recursivas de p usen los mismos argumentos: si la cabeza es $p(X_1,..., X_k)$, será correcto un antecedente como $p(V_1,..., V_k)$ siempre que exista una relación de orden parcial entre $<X_1,..., X_k>$ y $<V_1,..., V_k>$. Es una aproximación suficiente imponer que la relación de orden se cumpla en, al menos, uno de los pares $<X_i, V_i>$, siendo $1 \leq i \leq k$. Un literal q(A, B) es coherente con la ordenación parcial A<B; por ello, todo literal previo de la forma $q(V_m, V_n)$ establecerá un orden $V_m<V_n$.

2.5.4.3 Poda *alpha-beta*

Durante la evaluación de literales que introducen variables nuevas, se usa una especie de poda *alpha-beta*, para reducir la búsqueda: si se evalúan en primer lugar los literales más generales (los que introducen variables nuevas en la cláusula), pueden producirse situaciones en las que no sea necesario evaluar los literales más específicos (cuyos términos son variables ya usadas en la cláusula). Esto es así porque los literales no negados más específicos son satisfechos por menor número de tuplas positivas y, por tanto, su ganancia no puede superar cierta cota dada por el literal general del que proceden:

$$\text{Ganancia}(L_i^{esp}) \leq M_i^{++} \cdot \text{Info}(T_i) \qquad \text{[EC. 2.30]}$$

siendo M_i^{++} el número de tuplas positivas que satisfacen al literal L_i^{gen} (generalización de L_i^{esp}).

Este valor sólo será alcanzable cuando las M_i^{++} tuplas positivas (que satisfacen L_i^{gen}) y ninguna tupla negativa satisfagan al literal L_i^{esp}.

2.5.4.4 Uso de *literales determinados*

Dada una cláusula C inconsistente, con un conjunto de entrenamiento asociado T_i, se dice ([Quinlan, 91]) que L_i es un **literal determinado** (*determinate literal*) con respecto a esta cláusula cuando L_i introduce variables nuevas en C, y hay exactamente una extensión de cada tupla positiva de T_i en T_{i+1}, y no más de una extensión de cada tupla negativa. Por ello, si L_i es añadido a la cláusula, ninguna de las tuplas positivas será eliminada, y el nuevo conjunto T_{i+1} no tendrá más tuplas que T_i.

Si durante la búsqueda de nuevos literales no se encuentra ninguno con ganancia cercana a

$$\text{Ganancia}_{max} = \left| T_i^+ \right| \cdot \text{Info}(T_i)$$

la versión de FOIL descrita en [Quinlan, 91] añade a la nueva cláusula todos los literales determinados que encuentre en el espacio de búsqueda y repite la búsqueda. Esto hace que algunos de los literales añadidos no sean útiles, por lo que han de incorporarse mecanismos que eliminen los literales innecesarios al finalizar la construcción de cada cláusula. Puesto que con los literales determinados no se elimina ninguna tupla positiva y no aumenta la cardinalidad del conjunto de entrenamiento, el único coste de cálculo asociado se producirá por el aumento del espacio de búsqueda (debido a la introducción de nuevas variables) que es, precisamente, lo que se intenta conseguir.

Para evitar que los literales determinados encontrados en un ciclo den lugar a nuevos literales determinados y esto se repita de forma infinita, se limita el número de nuevas variables que pueden ser introducidas por estos literales dentro de una cláusula.

2.5.4.5 Definiciones incompletas y/o inconsistentes

En FOIL no se permite la construcción de cláusulas cuya complejidad supere a la de la enumeración explícita de las tuplas positivas cubiertas, considerando que dichas complejidades se miden por la longitud de codificación (número de bits necesarios para su codificación). Este heurístico también se conoce como principio de *Mínima Longitud de Descripción* de Rissanen (1983).

Si denominamos LDI(C) y LDE(C) a las longitudes de codificación de la descripción intensional (cláusula de Horn) y extensional (conjunto cubierto), respectivamente, de una cláusula C, podemos expresar el principio de Rissanen como:

$$\begin{aligned} &LDI(C) < LDE(C) \Rightarrow \text{Descripción Intensional (cláusula C)} \\ &LDE(C) < LDI(C) \Rightarrow \text{Descripción Extensional (enumeración tuplas)} \end{aligned} \qquad \text{[EC. 2.31]}$$

Estas longitudes de codificación se definen en el apartado 2.5.2.8 y 2.5.2.9.

FOIL sólo añade a la definición las cláusulas que, cumpliendo el principio de Rissanen, sean consistentes (precisión[1] = 100%) o bien, a pesar de ser *inconsistentes*, superen cierto umbral de precisión (por ej. el 85%). Cuando no es posible encontrar ninguna cláusula con estas condiciones, finaliza también la construcción de la definición, aunque ésta no cubra totalmente T^+, permitiendo así definiciones *incompletas*.

2.6 La incertidumbre en el conocimiento

Se pueden diferenciar dos etapas en la evolución del conocimiento ([Klir y Folger, 88]): un esfuerzo orientado a conocer aspectos del mundo y un posterior esfuerzo por conocer aspectos del propio conocimiento. Se puede suponer que ésta segunda etapa, en la que nos encontramos hoy en día, surge a consecuencia de los fallos de la primera, para delimitar el alcance y validez del conocimiento adquirido previamente. Nuestra preocupación no se centra en la mera adquisición de conocimiento, sino que, además, se intenta determinar en qué medida conocemos algo, qué grado de certeza podemos asignar a nuestro conocimiento. Hemos desviado nuestros problemas desde cómo manipular el mundo a cómo manipular el conocimiento (y la ignorancia) en sí mismo.

Se ha calificado a la nuestra como la *sociedad de la información*, y se destinan gran cantidad de recursos a la adquisición, manejo, procesado, selección, almacenamiento, distribución, protección, recopilación, análisis y clasificación de la información, para lo cual el ordenador resulta una herramienta de gran ayuda.

La gran cantidad de información de que disponemos, unida al grado de incertidumbre que lleva asociada, constituye la base de muchos de los problemas actuales: la complejidad.

1. Se define la *precisión* de una cláusula como la relación del número de tuplas positivas cubiertas respecto del total de tuplas cubiertas:

$$\text{precisión} = \left| T_C^+(C) \right| \; / \; \left| T_C(C) \right|$$

2.6.1 Tipos de incertidumbre y teorías matemáticas

El estudio de la información basado en su incertidumbre asociada[1] ha dado lugar a diferentes teorías matemáticas. La primera de ellas fue la conocida *teoría de la información* de Shannon (1948), construida a partir de la teoría clásica de conjuntos y de la teoría de la probabilidad.

Desde comienzos de los 80 se han realizado diferentes avances orientados a la construcción de una *teoría general de la información*. Dentro de ésta se incluyen, además de la teoría *clásica de conjuntos* y de la teoría de la *probabilidad*, otras como la teoría de *conjuntos borrosos*, la teoría de la *posibilidad* y la teoría de la *evidencia*.

Con las nuevas teorías se ha conseguido romper la relación única que existía entre incertidumbre y teoría de la probabilidad, y se ha pasado a considerar la incertidumbre en los términos mucho más genéricos de la teoría de conjuntos borrosos y de medidas borrosas. Además, ha quedado demostrado que la incertidumbre puede manifestarse en diferentes formas o, dicho de otro modo, que existen diferentes tipos de incertidumbre y que en la teoría de la probabilidad sólo se manifestaba una de ellas.

Los tres tipos de incertidumbre identificados con estas cinco teorías incluidas en la teoría general de la información son los siguientes:

- Borrosidad: resultante de la existencia de conjuntos borrosos, con límites vagamente definidos.
- Imprecisión o falta de especificidad: relacionada con el tamaño de conjuntos de alternativas.
- Discordia: producida por conflictos entre varios conjuntos de alternativas.

La imprecisión y la discordia pueden considerarse como diferentes modos de ambigüedad, asociando esta última con cualquier situación en la que no quede clara la alternativa correcta de un conjunto de ellas. Ésta puede deberse a una defectuosa caracterización de un objeto (imprecisión) o a distinciones conflictivas (discordias). Por otro lado, la borrosidad es diferente

1. Existen diferentes enfoques para medir la información ([Klir y Yuan, 95]):
   - Información como longitud de descripción: se basa en la teoría de la computación, y considera que la información representada por un objeto se puede medir por su longitud de descripción en algún lenguaje.
   - Información basada en la incertidumbre: considera que la información y la incertidumbre son conceptos muy relacionados, pues ésta se produce como resultado de alguna deficiencia en la información. La información obtenida por una acción puede calcularse como la reducción de incertidumbre que produce, asumiendo que ésta se puede medir dentro de una teoría matemática concreta.

   Nos centraremos en este último enfoque.

de la ambigüedad, y se produce cuando existen conceptos cuyos límites no están perfectamente determinados.

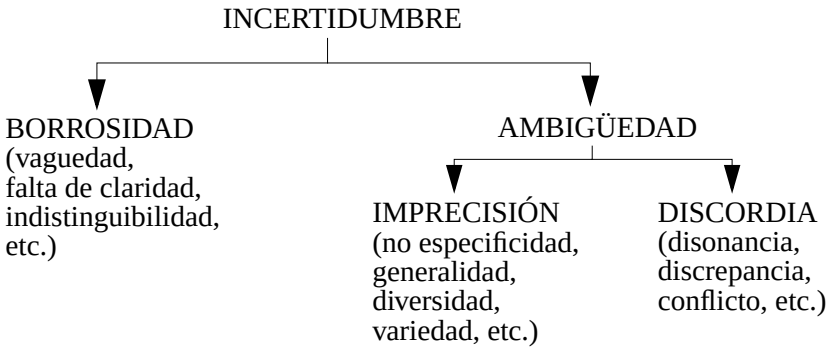

Figura 2-4: Tipos de incertidumbre

Es posible que, en el futuro, se identifiquen otros tipos de incertidumbre, como resultado de la investigación de nuevas teorías sobre la incertidumbre.

Dentro de cada una de las teorías actualmente existentes es posible medir diferentes tipos de incertidumbre, como se muestra en la tabla 2-2 ([Klir y Yuan, 95]) para conjuntos finitos.

Tabla 2-2: Medidas de incertidumbre para conjuntos finitos

Teoría sobre incertidumbre	**Tipo de Incertidumbre**		
	Borrosidad	**Imprecisión**	**Discordia**
Teoría clásica de conjuntos	-	$U(A) = \log_2\|A\|$	-
Teoría de la probabilidad	-	-	$H(m) = -\sum_{x \in X} m(\{x\})\log_2(m(\{x\}))$
Teoría de conjuntos borrosos	$f(A) = \sum_{x \in X} [1 - \|2A(x) - 1\|]$	$U(A) = \frac{1}{h(A)} \int_0^{h(A)} \log_2\|{}^{\alpha}A\| d\alpha$	-
Teoría de la posibilidad	-	$U(r) = \sum_{i=2}^{n} r_i \cdot \log_2 \frac{i}{i-1}$	$S(r) = \sum_{i=2}^{n} (r_i - r_{i+1}) \cdot \log_2 \frac{i}{\sum_{j=1}^{i} r_j}$
Teoría de la evidencia	-	$N(m) = \sum_{A \in F} m(A) \cdot \log_2\|A\|$	$S(m) = -\sum_{A \in F} m(A)\log_2 \sum_{B \in F} m(B)\frac{\|A \cap B\|}{\|A\|}$

En el caso de conjuntos infinitos, no existen ecuaciones aplicables para algunas de las medidas de incertidumbre, en particular en las teorías de la probabilidad y de la evidencia. Otras, como la borrosidad en la teoría de los conjuntos borrosos, son fácilmente obtenidas a partir de la expresión para conjuntos finitos; en el caso de un conjunto infinito pero acotado en [a, b], este valor será:

$$f(A) = \int_a^b (1 - |2A(x) - 1|)dx = b - a - \int_a^b |2A(x) - 1|dx$$

2.6.2 Conjuntos borrosos

El concepto de conjunto borroso fue introducido por [Zadeh, 65], motivado por su interés por el análisis de sistemas complejos de control. En los conjuntos borrosos desaparece la drástica distinción entre miembros y no miembros de la teoría clásica de conjuntos, permitiendo que los elementos de un subconjunto A del conjunto universal U tengan asociado un grado de pertenencia comprendido en el intervalo [0, 1]. La función de pertenencia μ sustituye a la función característica de los conjuntos clásicos:

$$\mu_A : U \rightarrow [0, 1]$$

Por este motivo, los conjuntos borrosos constituyen un método natural de representar la imprecisión y subjetividad propias de la actividad humana.

En el apartado A.1 pueden verse diferentes conceptos relacionados con los conjuntos borrosos, desde la definición del conjunto de pertenencia hasta la definición de diferentes operaciones aplicables sobre conjuntos borrosos, algunas de ellas también existentes para conjuntos clásicos (intersección, unión, complementación, etc.) y otras más específicas.

Del mismo modo que la teoría clásica de conjuntos y la lógica de proposiciones son isomorfas entre sí (ambas tienen estructura de álgebra de Boole), [Zadeh, 75] propone una lógica borrosa isomorfa con la teoría de conjuntos borrosos.

2.6.3 Medidas borrosas

Conviene distinguir entre dos conceptos parecidos que pueden llevar a confusión: los *conjuntos borrosos* y las *medidas borrosas*.

Una medida borrosa representa incertidumbre en la pertenencia de un elemento a un conjunto, pero ésta no se debe a que el conjunto en sí mismo tenga naturaleza imprecisa o borrosa, sino a que no se conoce con precisión (aunque podría conocerse) en qué medida pertenece cada elemento a dicho conjunto. Las medidas borrosas se pueden realizar sobre conjuntos ordinarios (no borrosos). Por el contrario, los conjuntos borrosos tienen límites imprecisos, y por su propia definición no se puede determinar con precisión el grado de pertenencia de los elementos.

Por ejemplo:

> Sean tres conjuntos ordinarios, con las personas de edad menor que 20 años, entre 20 y 40 años, y mayor de 40 años. Estos conjuntos son ordinarios, pues no existe ninguna incertidumbre en sus límites. Sin embargo, para una persona concreta puede que no conozcamos su edad exacta sino sólo una estimación de la misma y, por tanto, no podamos clasificarla con total seguridad en ninguno de los conjuntos definidos. En ese caso, asignaremos un cierto grado de certeza o incertidumbre a su pertenencia a cada conjunto; esta representación de la incertidumbre se conoce como medida borrosa.

Por otro lado, también podríamos definir los conjuntos de las personas jóvenes y las viejas. En este caso, tendríamos conjuntos borrosos, pues los conceptos de joven y viejo no pueden separarse de forma rigurosa, sino que sus límites son difusos. Aunque conociéramos con precisión la edad de una persona, ésta tendría asociada un grado de pertenencia a cada uno de estos conjuntos borrosos.

Una medida borrosa se define ([Klir y Yuan, 95]) como una función de la forma

$$g: C \rightarrow [0, 1]$$

siendo *C* una familia no vacía de subconjuntos dentro de un conjunto universal U. La función *g* debe cumplir:

1. $g(\varnothing) = 0, g(U) = 1$ (valores de los extremos)

2. $A \subseteq B \Rightarrow g(A) \leq g(B), \forall\ A, B \in C$ (monotonicidad)

3. $\bigcup_{i=1}^{\infty} A_i \in C \Rightarrow \lim_{i \to \infty} g(A_i) = g\left(\bigcup_{i=1}^{\infty} A_i\right)$ (continuidad desde abajo)

4. $\bigcap_{i=1}^{\infty} A_i \in C \Rightarrow \lim_{i \to \infty} g(A_i) = g\left(\bigcap_{i=1}^{\infty} A_i\right)$ (continuidad desde arriba)

Puesto que para cualquier par de conjuntos A y B se cumple que $A \cap B \subseteq A$ y $A \cap B \subseteq B$, aplicando la condición de monotonicidad se comprueba que

$$g(A \cap B) \leq \min(g(A), g(B)) \qquad \text{[EC. 2.32]}$$

y, de modo análogo, se demuestra que

$$g(A \cup B) \geq \max(g(A), g(B)) \qquad \text{[EC. 2.33]}$$

Dentro de la teoría de medidas borrosas se incluyen, entre otras, la teoría de la evidencia, la teoría de la posibilidad y la teoría de la probabilidad (figura 2-5). La teoría de medidas borrosas (así como cualquiera de sus ramas) puede ser aplicada tanto sobre conjuntos ordinarios como sobre conjuntos borrosos.

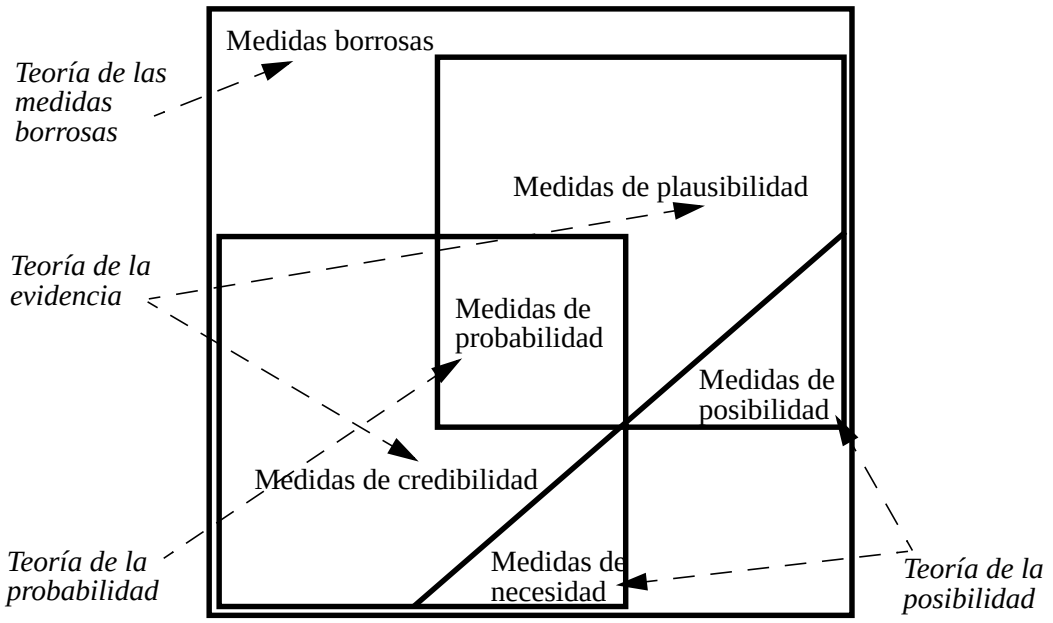

Figura 2-5: Relación entre las diversas clases de medidas borrosas

2.6.3.1 Teoría de la evidencia

La teoría de la evidencia, de Dempster-Shafer, se basa en dos medidas duales no aditivas: medidas de credibilidad (*belief measures*) y medidas de plausibilidad (*plausibility measures*).

- Dado un conjunto universal U finito, una *medida de credibilidad* es una función de la forma:

  $$Bel: P(U) \rightarrow [0, 1]$$

  siendo $P(U)$ el conjunto de subconjuntos de U (conjunto potencia). La función *Bel* debe cumplir:

  1. $Bel(\varnothing) = 0$, $Bel(U) = 1$
  2. $Bel(A_1 \cup ... \cup A_n) \geq$

     $$\geq \sum_j Bel(A_j) - \sum_{j<k} Bel(A_j \cap A_k) + \ldots + (-1)^{n+1} Bel(A_1 \cap \ldots \cap A_n)$$

  Esta segunda expresión también se conoce como condición de *superaditividad*. Si U es infinito, entonces se requiere también que *Bel* sea continua desde arriba.

  Es inmediato deducir que

  $$Bel(A) + Bel(\overline{A}) \leq 1. \qquad \text{[EC. 2.34]}$$

  Para cualquier conjunto $A \in P(U)$, $Bel(A)$ se interpreta como el grado de credibilidad (basado en la evidencia disponible) de que un elemento dado de U pertenezca al conjunto A.

- Asociada a toda medida de credibilidad hay una *medida de plausibilidad, Pl*, definida por la ecuación:

$$Pl(A) = 1 - Bel(\overline{A}) \qquad \text{[EC. 2.35]}$$

Las medidas de plausibilidad y de credibilidad son duales entre sí. Se puede comprobar que

$$Pl(A) + Pl(\overline{A}) \geq 1 \qquad \text{[EC. 2.36]}$$

Para cualquier conjunto $A \in P(U)$, $Pl(A)$ se interpreta como el grado de credibilidad (basado en la evidencia disponible) de que un elemento dado de U pertenezca al conjunto A o a cualquier subconjunto cuya intersección con A sea no vacía. Por tanto,

$$Pl(A) \geq Bel(A), \qquad \forall A \in P(U) \qquad \text{[EC. 2.37]}$$

Las medidas de credibilidad y plausibilidad pueden ser caracterizadas por una función:

$$m: P(U) \rightarrow [0, 1]$$

tal que

$$m(\varnothing) = 0,$$

y $$\sum_{A \in P(U)} m(A) = 1$$

Esta función se denomina *asignación de probabilidad básica*. Dada una asignación básica m, los valores de credibilidad y plausibilidad se determinan de forma única:

$$Bel(A) = \sum_{B|(B \subseteq A)} m(B) \qquad \text{[EC. 2.38]}$$

$$Pl(A) = \sum_{B|(A \cap B \neq \varnothing)} m(B) \qquad \text{[EC. 2.39]}$$

2.6.3.2 Teoría de la posibilidad

Un caso particular de teoría de la evidencia es la *teoría de la posibilidad*. La teoría de la posibilidad maneja únicamente conjuntos anidados; dentro de ella caben mencionar las medidas de necesidad y de posibilidad, como casos particulares de las medidas de credibilidad y plausibilidad respectivamente.

Las medidas de necesidad (*Nec*) y posibilidad (*Pos*) cumplen:

$$Nec(A \cap B) = \min(Nec(A), Nec(B)) \qquad \text{[EC. 2.40]}$$

$$Pos(A \cup B) = \max(Pos(A), Pos(B)) \qquad \text{[EC. 2.41]}$$

que puede comprobarse que son casos particulares de las ecuaciones [EC. 2.32] y [EC. 2.33].

Cada medida de posibilidad *Pos* en el conjunto potencia $P(U)$ se define de forma unívoca por una *función distribución de posibilidad*:

$$r: U \rightarrow [0, 1]$$

mediante la fórmula

$$Pos(A) = \max_{x \in A} \{ r(x) \}, \forall A \in P(U) \qquad \text{[EC. 2.42]}$$

Otra forma alternativa de formular la teoría de la posibilidad es en términos de conjuntos borrosos. Las medidas de posibilidad están directamente conectadas con los conjuntos borrosos a través de las funciones de distribución de posibilidad:

Consideremos el conjunto borroso F, formado por los posibles valores de una variable V, y la expresión V = v, con el valor $v \in F$, para indicar que el valor de V es v. Entonces, para un valor concreto v, se puede interpretar $\mu_F(v)$ como el grado de compatibilidad de v con el concepto descrito por F; por otro lado, dada la proposición "V es F", es más sencillo interpretar $\mu_F(v)$ como el grado de posibilidad de la condición V = v. Es decir, para cada valor v se cumple que la posibilidad de V = v es numéricamente igual al grado de pertenencia de v al conjunto F:

$$r_F(v) = \mu_F(v), \quad \forall\, v \qquad \text{[EC. 2.43]}$$

Por tanto, la función distribución de posibilidad puede considerarse como una función de pertenencia normalizada (con al menos un grado de pertenencia igual a 1)[1].

2.6.3.3 Teoría de la probabilidad

Una medida de probabilidad, *Pro*, debe satisfacer la condición de *aditividad*

$$Pro(A \cup B) = Pro(A) + Pro(B), \qquad \forall A,B \in P(U), \text{ t.q. } A \cap B = \varnothing \qquad \text{[EC. 2.44]}$$

que no es más que un caso particular (más restrictivo) de la condición de superaditividad. Las medidas de probabilidad son un tipo especial de medidas de credibilidad, como expresa el siguiente teorema: "una medida de credibilidad *Bel* en un conjunto potencia *P*(U) es una medida de probabilidad si y sólo si la función asignación de probabilidad básica m es tal que: $m(\{x\}) = Bel(\{x\})$ y $m(A) = 0$, para todo subconjunto A de U con más de un elemento".

Bajo estas condiciones, las medidas de credibilidad y plausibilidad coinciden:

$$Bel(A) = Pl(A) = \sum_{x \in A} m(\{x\}) \qquad \text{[EC. 2.45]}$$

No debe confundirse la probabilidad con el grado de pertenencia a un conjunto borroso (a diferencia de lo que ocurre con la posibilidad). Una forma rápida de comprobar que el grado de pertenencia de un elemento a un conjunto no se corresponde con su probabilidad de pertenecer a dicho conjunto, es viendo que los grados de pertenencia a los diferentes conjuntos borrosos no tienen por qué sumar uno y, sin embargo, la suma de las probabilidades de pertenecer a diferentes conjuntos debe ser igual a la unidad.

Por tanto, la teoría de la probabilidad, al igual que la teoría de la posibilidad, es un caso particular de la teoría de la evidencia, que a su vez está incluida en la teoría más general de las medidas borrosas (figura 2-5).

1. En el caso de que el conjunto F, del que se deriva el valor r_F, no estuviera normalizado, no se podría aplicar la función de asignación de probabilidad básica, como se demuestra en [Klir y Yuan, 95].

2.6.4 La lógica borrosa y el mundo real

El método que empleamos habitualmente para manejar con éxito la complejidad del mundo real consiste en simplificarla, buscando un compromiso entre la información de que disponemos y la incertidumbre que podemos permitir. Una opción es incrementar la cantidad de incertidumbre permitida, sacrificando parte de la información precisa en favor de una información más vaga pero más robusta.

Por ejemplo, en vez de describir el tiempo que hace hoy en términos de un porcentaje exacto de la porción de cielo cubierto por las nubes (que sería demasiado complejo), podemos decir simplemente que está soleado, lo cual reduce la complejidad añadiendo incertidumbre, aunque es una descripción mucho más útil. El lenguaje humano es típicamente impreciso o borroso y, sin embargo, no por ello carece de significación. El término "soleado" introduce cierto grado de imprecisión, dado que no lo usamos para indicar únicamente un 0% de nubes en el cielo, sino también un 10% de nubes, por ejemplo, o incluso un 20% de nubes. Por otro lado, sería inaceptable usarlo para un cielo cubierto de nubes en su totalidad, ni siquiera para un 80% de nubes. ¿Dónde establecemos entonces el límite entre un día soleado y un día cubierto?

Debemos aceptar que "soleado" introduce cierta imprecisión y que existe una separación vaga o borrosa (dependiente del porcentaje de cielo cubierto) entre los días soleados y los no soleados. Éste es precisamente el concepto de conjunto borroso o difuso (*fuzzy set*, [Zadeh, 65]), que no es más que una generalización del tradicional conjunto ordinario (*crisp set*).

En [Fernández y Sáez-Vacas, 87] se defiende que la lógica borrosa va más allá que las lógicas multivaloradas con infinitos valores de verdad, porque no sólo considera que hay una infinidad de valores semánticos entre "verdadero" y "falso", sino que también tiene en cuenta que esos mismos valores de verdad son imprecisos. Así, por ejemplo, en lógica multivalorada se podría asignar el valor de verdad 0.7 a la sentencia "este párrafo es de difícil comprensión", pero no nos permitiría hacer inferencias imprecisas como "si alguien encuentra muy difícil comprender un párrafo, es probable que abandone su lectura". La lógica multivalorada no nos lo permite porque intervienen matices (relaciones entre "difícil", "muy difícil", "probable") imposibles de abordar con la simple extensión del conjunto de valores de verdad.

Ese tipo de inferencia imprecisa es el que pretende abordar la lógica borrosa, y para ello parte de la reconsideración del concepto matemático de *conjunto*. En la realidad pueden encontrarse infinidad de situaciones en las que resulta difícil determinar la pertenencia o no de un elemento a un conjunto:

- conjunto de números mucho mayores que 100
- conjunto de personas pobres
- conjunto de mujeres guapas, etc.

Por tanto, en la lógica borrosa encontramos un método natural de representar la información del mundo real, gracias a la similitud existente entre los conceptos humanos y los conjuntos borrosos.

Podemos encontrar otros muchos ejemplos en los que el lenguaje humano introduce imprecisión, sin perder por ello significado. Es más, las relaciones imprecisas existentes entre el hombre y su entorno son mucho más significativas para el hombre que otro tipo de relaciones que podrían medirse con más precisión. Por ejemplo, cualquiera puede entender una orden imprecisa como "levanta ligera y lentamente el pie del embrague", mientras que es difícil realizar otras como "levanta el pie 10˚ a una velocidad de 2˚30' por segundo".

De modo similar, en cualquier otro sistema complejo distinto del hombre, como las sociedades, una central térmica, etc. que se pretenda analizar, será imprescindible introducir en los modelos

esta imprecisión y subjetividad utilizadas desde el punto de vista humano. Además, en este tipo de sistemas complejos, se cumple que *significación* y *precisión* son términos incompatibles ([Zadeh, 73]). Los resultados muy precisos suelen tener poco significado, y lo que interesa más bien son resultados cualitativos.

2.6.4.1 Sistemas expertos borrosos

En la mayoría de los actuales sistemas expertos se presenta este problema: es necesario representar en la máquina los conocimientos y procedimientos inciertos e imprecisos que utilizan los expertos humanos para resolver problemas, y para ello se adoptan normalmente técnicas *ad hoc*. Pero existen intentos teóricos para introducir en la lógica formal la imprecisión y subjetividad característica de la actividad humana, y puede esperarse que en el futuro el diseño de sistemas expertos se base en tales teorías.

Parece claro que el conocimiento humano se basa en apreciaciones tanto de naturaleza probabilística (basada en la repetición de un fenómeno) como de tipo subjetivo o particular, que hacen necesario trabajar simultáneamente con probabilidades y con posibilidades para modelar de forma adecuada este conocimiento. Por ejemplo, en medicina es frecuente encontrar reglas probabilísticas como:

"la medicina X provoca vómitos en un 2 por 1000 de los casos"

basadas en un estudio previo de pacientes a los que se ha suministrado dicha medicina. Sin embargo, reglas como:

"una persona es atractiva si tiene buena presencia y es divertida e inteligente"

están basadas en apreciaciones más o menos subjetivas sobre la atracción de las personas. Por tanto, es deseable utilizar un modelo en el que se puedan manipular conjuntamente ambos tipos de incertidumbre, como se propone en [Pons et al., 94].

En un sistema experto borroso, el proceso de inferencia es una combinación de cuatro subprocesos ([Horstkotte, 96]):

1. Borrosificación: proceso por el que se aplican las funciones de pertenencia definidas en las variables de entrada sobre los valores reales de los atributos, para determinar el grado de verdad de las premisas de cada regla. La determinación de los grados de pertenencia suele realizarse mediante métodos empíricos, como se describe en el apartado A.2.

2. Inferencia: a partir del valor de verdad calculado para las premisas de cada regla se calcula el de la conclusión de la misma. Este resultado es un subconjunto borroso aplicable a cada variable de salida de cada regla. Es frecuente hablar de inferencia de tipo MAX-MIN o de tipo SUMA-PRODUCTO, que deben interpretarse como la combinación de una composición MAX con una inferencia MIN, o una composición SUMA con una inferencia PRODUCTO, usando esta división en subprocesos.

3. Composición: se combinan los subconjuntos borrosos obtenidos para las variables de salida en un único subconjunto borroso para cada variable. Para ello se suele usar una composición de tipo MAX o de tipo SUMA.

4. *Desborrosificación*: A veces es útil examinar los conjuntos borrosos resultantes del proceso de composición, aunque otras veces se necesita convertir el valor borroso en un valor no borroso, para lo que se aplica un proceso de desborrosificación. Dos de las técnicas más usadas de desborrosificación son la del CENTROIDE y el valor MÁXIMO, aunque existen diferentes variantes de ellas.

En un plano de mayor abstracción cabe mencionar el trabajo de [González, 89], en el que se propone una nueva arquitectura para la construcción de un entorno de desarrollo de sistemas expertos. Esta arquitectura está diseñada para ser utilizada en dominios en los que la incertidumbre es un factor esencial a considerar. Mediante esta arquitectura se pretende facilitar la aplicación de distintas metodologías de razonamiento aproximado, entre ellas la basada en conjuntos y lógica borrosa, el enfoque probabilista, etc.

2.6.4.2 Bases de datos relacionales borrosas

El modelo relacional de base de datos, propuesto por Codd a principios de los años 70 ([Codd, 70]), es el predominante dentro de las bases de datos que existen en la actualidad. Quizá por ello, la mayoría de las bases de datos borrosas que se describen en la literatura se basan también en dicho modelo relacional.

a) Modelo de Buckles y Petry

En [Klir y Yuan, 95] se describe un modelo de base de datos borrosa (propuesto por Buckles y Petry en 1982), que incluye, como caso particular, al clásico modelo relacional. Las únicas diferencias incluidas en este nuevo modelo son dos:

- Los valores que toman los atributos de las tuplas son, en general, subconjuntos de los correspondientes dominios, y no sólo valores atómicos.
- Se define una relación de similitud dentro de cada dominio.

La primera de estas diferencias permite, por ejemplo, representar diferentes opiniones de varios expertos, mediante un conjunto que las incluya a todas ellas dentro de una tupla. Por ejemplo, una relación que describa diferentes mercados para una empresa podría ser:

Relación: MERCADOS

REGIÓN	TAMAÑO	POTENCIAL
Este	grande	bueno
Oeste	(grande, medio)	(regular, bueno)
Sur	pequeño	(bueno, excelente)

La segunda aportación de este modelo se basa en la suposición de que en el modelo relacional clásico existe una relación de equivalencia para cada dominio, de modo que se establecen clases de equivalencia (que coinciden con cada uno de los valores atómicos) en cada dominio. La generalización de esta idea conduce a la definición de relaciones de equivalencia borrosas (o relaciones de similitud) para cada dominio. Por ejemplo, para el dominio correspondiente al atributo “potencial” de la anterior relación, se pueden definir la siguiente relación de similitud, para los valores “excelente” (E), “bueno” (B), “regular” (R) y “malo” (M):

	E	**B**	**R**	**M**
E	1	0.8	0.4	0
B	0.8	1	0.5	0.1

R	0.4	0.5	1	0.5
M	0	0.1	0.5	1

El álgebra relacional utilizada en este modelo de base de datos borrosa incluye las mismas operaciones que en el caso ordinario y, además, permite definir el mínimo grado de similitud aceptable entre elementos de cada dominio, para las operaciones de consulta a la base de datos.

b) Modelo de Umano

Otra interesante extensión del modelo relacional para bases de datos borrosas se debe a Umano (1982). En este modelo, las relaciones que se utilizan son borrosas, ya que sus tuplas tienen un grado de pertenencia asociado.

También se introduce incertidumbre en las consultas a la base de datos, mediante la utilización de conjuntos borrosos para los dominios implicados en las mismas. De este modo, a partir de una relación ordinaria de la base de datos, se puede obtener una nueva relación borrosa añadiendo a cada tupla un grado de pertenencia dado por su similitud con el valor borroso utilizado en la consulta.

La selección de tuplas para una consulta dependerá del umbral de similitud que se defina.

c) Modelo de Medina

En [Medina et al., 94] se propone una generalización que incluye los dos modelos anteriores de bases de datos relacionales borrosas. Para ello se utilizan nuevas estructuras de datos y un álgebra relacional modificada.

Los nuevos tipos de datos dentro de este modelo tienen como objetivo la representación de información borrosa, mediante distribuciones de posibilidad, números borrosos y etiquetas lingüísticas. Mediante estos nuevos tipos de datos se construyen las relaciones de la base de datos, denominadas “relaciones borrosas generalizadas”.

Este modelo describe también un “Álgebra Relacional Borrosa Generalizada”, con operaciones como la unión, intersección, diferencia producto cartesiano, proyección, *join* y selección, aplicables sobre las nuevas relaciones borrosas generalizadas.

En este modelo se pueden representar, por ejemplo, relaciones como la siguiente:

NOMBRE	DIRECCIÓN	EDAD	PRODUCT.	SALARIO
Luis	Recogidas	&.8/30,1/31	Buena	110000
Antonio	R.Católicos	Media	Normal	100000
J.Carlos	C.Ronda	#28	Excelente	150000

donde el símbolo & representa una distribución de posibilidad, y # significa “aproximadamente”; los valores “media”, “buena”, “normal” y “excelente” son etiquetas lingüísticas.

Y se pueden realizar consultas borrosas como: “dados el nombre, edad, productividad y salario, así como un grado de satisfacción mínimo para cada atributo, seleccionar las tuplas en las que se cumpla que la productividad es buena (en grado mayor que 0.9) y el salario es alto (en grado menor que 0.7)”.

Capítulo 3:

EVALUACIÓN DE LITERALES (MEJORAS DEL SISTEMA FOIL)

3.1 Introducción

Una de las principales deficiencias de FOIL es que la búsqueda que realiza dentro del espacio de literales candidatos podría calificarse de *corta vista*, ya que selecciona el mejor de los literales explorados, sin tener en cuenta otras posibilidades que podrían dar mejores resultados globales en pasos posteriores. Este método de búsqueda es sencillo, pero los resultados que genera no son siempre lo suficientemente buenos, debido a los *máximos locales* (tan conocidos en cualquier sistema de optimización) que se producen en muchas ocasiones. Con frecuencia las definiciones que se construyen son complicadas, con más cláusulas o con cláusulas más largas de lo necesario, e incluso, a veces, no se encuentra ninguna definición completa y consistente aunque realmente exista.

Durante el proceso de construcción de definiciones tiene gran importancia no sólo el algoritmo de búsqueda (de *corta vista* en FOIL), sino también la función de evaluación de literales que se emplee (*ganancia* en FOIL). La forma de esta función de evaluación puede determinar, precisamente, la existencia o no de máximos locales.

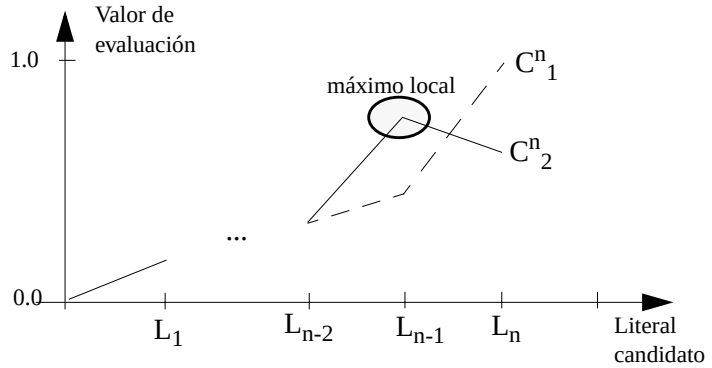

Figura 3-1: Máximos locales en la evaluación de literales

Aunque FOIL incorpora algunos sencillos mecanismos de retroceso (*backtracking*), éstos sólo se disparan cuando, durante la construcción de una cláusula, se llega a un estado sin soluciones (sin literales que mejoren la cláusula en construcción) y en algún paso previo se haya realizado un selección "arbitraria" de un literal (cuando existen varios con un valor de *ganancia* igual o similar). Estos mecanismos no son suficientes para evitar los máximos locales, y se producen errores durante la construcción de las cláusulas.

Cabe mencionar aquí otro algoritmo de Quinlan: ID3 ([Quinlan, 86]), que, al igual que FOIL, realiza una búsqueda heurística basada en una medida de la ganancia de información (reducción de entropía).

El objetivo de ID3 (construcción de un árbol de decisión con un lenguaje que puede asimilarse a la lógica de proposiciones extendida, apartado 2.3.1.1) es diferente del de FOIL, así como la función *ganancia* empleada. Sin embargo, también se han detectado algunas deficiencias en esta última. En el caso de ID3, la medida de *ganancia* utilizada favorece a los atributos con más valores. Quinlan propuso dividirla por un factor de ganancia (*Gain Ratio*), para mitigar este efecto, aunque esto presenta otras nuevas limitaciones. En [López de Mántaras, 91] se propone una nueva función para seleccionar atributos, dentro de ID3, basándose en una medida de la *distancia* entre las particiones (definidas por los valores de cada atributo), con la que se mejoran los resultados de ID3.

En los siguientes apartados se analizarán diferentes errores producidos por la *ganancia* utilizada en FOIL, que pueden dar lugar a máximos locales, y se propondrá una nueva función de evaluación para solventarlos. Aunque esta nueva función, basada en medidas del *interés* de una regla, no garantiza la eliminación de máximos locales, al menos corrige las deficiencias detectadas en la medida de la *ganancia*, mejorando los resultados en diferentes ejemplos.

Otro método de abordar el problema de los máximos locales consiste, como ya se ha mencionado, en la exploración del espacio de búsqueda mediante nuevos algoritmos, aunque esta vía se deja abierta como futura línea de investigación (apartado 6.2.1.2).

3.2 Errores de la función de evaluación *Ganancia*

El valor de la función *Ganancia* (apartado 2.5.4.1) depende únicamente de la información de los conjuntos T_i y T_{i+1} (función de la concentración de tuplas positivas) y del número de tuplas positivas de T_i que satisfacen al literal candidato L_i (es decir, N_i^{++}). Esto hace que, en algunas ocasiones, este valor no se corresponda con la *bondad* de un literal, considerando que ésta debe reflejar la capacidad de la definición para cubrir de forma completa y consistente el conjunto de entrenamiento inicial; además, deben preferirse las definiciones sencillas, esto es, de mínima longitud (suponiendo que la longitud, definida en ecuación [EC. 2.26], puede ser una medida de la complejidad).

Hemos detectado tres tipos de errores que pueden ocurrir al evaluar los literales con la función *Ganancia*, tal y como se describe a continuación; sus consecuencias pueden ser las mismas: desde un simple incremento en la complejidad de la definición hasta la construcción de definiciones inexactas. Algunos resultados preliminares ya fueron presentados en [Gómez, 92].

3.2.1 Error tipo I: *Ganancia insensible a tuplas negativas cubiertas*

El valor de la función *Ganancia* es *insensible al número de tuplas negativas cubiertas*. Supongamos la situación descrita en la figura 3-2: tenemos el conjunto T_i (con tres tuplas positivas: p1, p2 y p3, y dos negativas: n1 y n2), y estamos evaluando dos literales candidatos L_i^a y L_i^b; ambos literales cubren las mismas tuplas positivas (p1 y p2), y generan nuevos conjuntos con igual información (con cuatro tuplas positivas y dos negativas). La única diferencia es que sólo una tupla negativa de T_i satisface L_i^b, mientras que son dos las tuplas negativas que satisfacen L_i^a.

Sería lógico suponer que, a igualdad de tuplas positivas cubiertas, deberían favorecerse los literales satisfechos por menos tuplas negativas (es decir, L_i^b); sin embargo, la función *Ganancia* no diferencia entre ambos en situaciones como la del ejemplo.

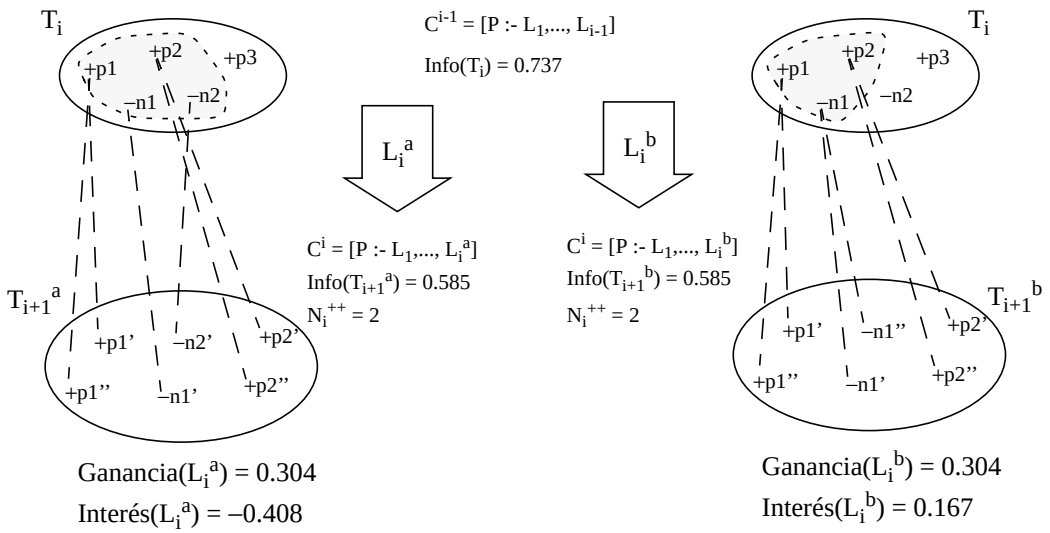

Figura 3-2: Error tipo I

Una elección incorrecta en situaciones de este tipo puede no tener ninguna consecuencia en el resultado final del algoritmo, aunque, en el caso peor, puede dar lugar a una definición más larga de lo necesario o, incluso, una definición incompleta (si se llegase a superar la longitud de descripción extensional). Por ello, una cláusula inconsistente en la que haya tenido lugar algún error de este tipo tendrá mayor probabilidad de ser rechazada, finalizando entonces la construcción con una definición incompleta. Pero, incluso si eligiendo el literal que genera la cláusula más precisa (L_i^b) tampoco se evita la existencia de una definición incompleta, al menos ésta cubrirá menos tuplas negativas del conjunto inicial, por lo que el error producido será menor.

La función *Interés* (que estudiaremos más adelante) sí tiene en cuenta el número de tuplas negativas cubiertas, de modo que puede diferenciar entre ambos literales y elegir correctamente el que proporciona mayor precisión.

3.2.2 Error tipo II: *Ganancia no proporcional a tuplas positivas cubiertas*

En algunas ocasiones la función *Ganancia no* es *proporcional al número de tuplas positivas cubiertas*. Esto ocurre cuando el grado en que se expanden las tuplas del conjunto local difiere en

gran medida para diferentes literales candidatos. Podemos ver esta situación en el ejemplo descrito en la figura 3-3:

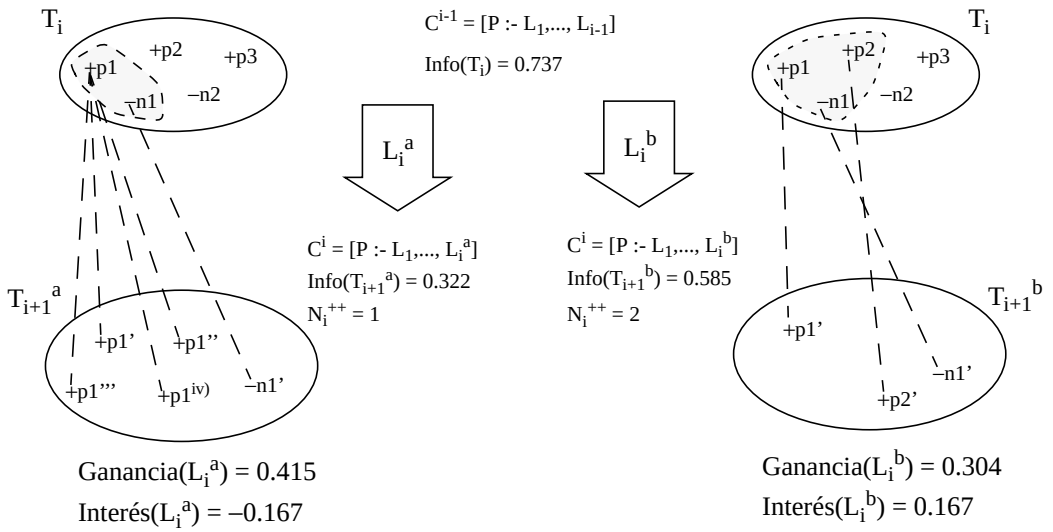

Figura 3-3: Error tipo II

- uno de los literales (L_i^a) genera un nuevo conjunto (T_{i+1}^a) en el que una de las tuplas cubiertas (+p1) se expande en cuatro, lo que disminuye la información del conjunto T_{i+1}^a, y aumenta la *Ganancia* resultante;
- el otro literal (L_i^b), por el contrario, aunque incluye en el conjunto cubierto una tupla positiva más (+p2) que el primero, expande sus tuplas en menor medida (cada tupla de T_i genera sólo una en T_{i+1}^b), lo que hace que la *Ganancia* sea inferior.

Usando como heurístico de evaluación la función *Ganancia*, se pueden elegir literales (como L_i^a en este ejemplo) que cubran menos tuplas positivas (e igual o mayor número de tuplas negativas) que otros candidatos, favoreciendo así la construcción de cláusulas demasiado específicas, es decir, con conjuntos cubiertos pequeños. Esto obliga a que las definiciones construidas necesiten más cláusulas para cubrir completamente el conjunto de ejemplos de entrenamiento, lo que repercute en un incremento en la complejidad de las definiciones (medida por su longitud de codificación) y, posiblemente, también un incremento en el coste asociado a su construcción (coste de la búsqueda de literales). Por ello, situaciones como la descrita pueden considerarse como errores de evaluación de literales.

Pero el mayor problema aparece, de nuevo, cuando la cláusula supera la longitud de codificación máxima permitida y, consecuentemente, se pone en marcha el heurístico que permite la existencia de definiciones inexactas. En este caso, la cláusula construida con el literal erróneo tendrá menor precisión que la otra alternativa y, por tanto, menor probabilidad de ser añadida a la definición correspondiente (en el supuesto de que se permita su inconsistencia), tal y como vimos en el apartado 2.5.4.5. Por el contrario, si la elección del literal fuera la otra (usando un heurístico como el *Interés*, tal y como veremos), y también finalizara la construcción de la cláusula (inconsistente), sería más probable superar el umbral de precisión necesario, construyéndose así una definición completa (o, al menos, que cubra más tuplas positivas).

3.2.3 Error tipo III: *argumentos no representan al conjunto cubierto*

No debemos olvidar que las tuplas positivas que se quieren cubrir con la definición construida no son las del conjunto local T_i, sino las del conjunto inicial T^+. Por tanto, al aplicar el heurístico de evaluación *Ganancia* sobre los argumentos medidos en los sucesivos T_i, estamos introduciendo errores en la evaluación de los literales, ya que los resultados calculados sólo reflejan de manera indirecta el comportamiento real de los candidatos.

Dos literales que cubran un mismo número de tuplas positivas y negativas en uno de los conjunto intermedios T_i deberían tener diferente valor de evaluación si los correspondientes conjuntos cubiertos (medidos sobre el conjunto inicial T_1) no coinciden. En la función *Ganancia*, tal como se ha definido, esto no es así, por lo que se puede producir un error que hemos denominado de tipo III, al medir *argumentos que no representan a los del conjunto cubierto*. Podemos ver de forma gráfica este problema en la figura 3-4:

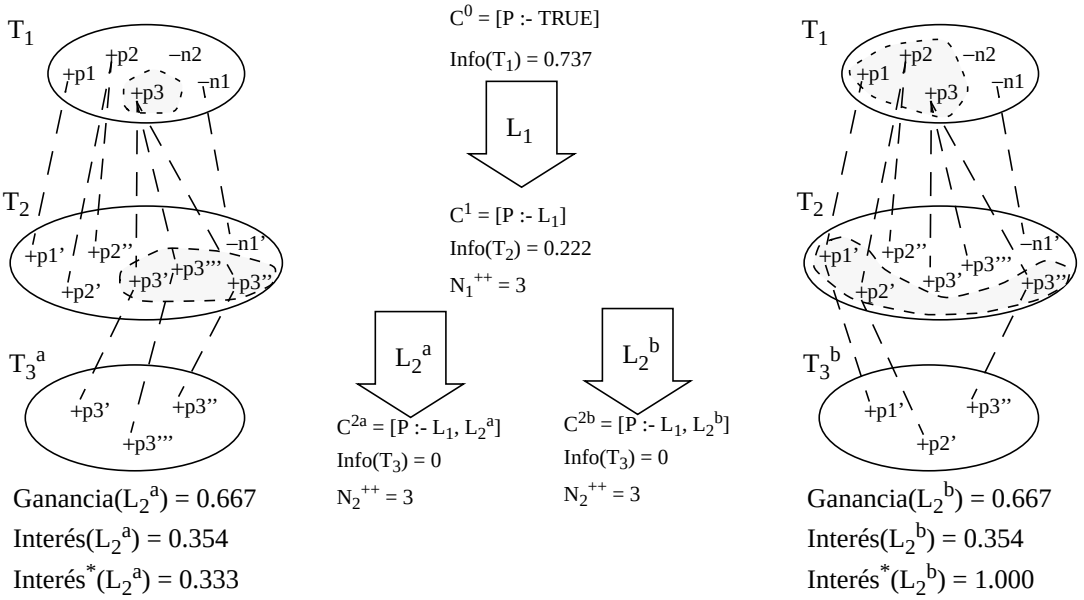

Figura 3-4: Error tipo III

- Uno de los literales (L_2^a) cubre a tres tuplas positivas del conjunto intermedio (T_2) y a ninguna negativa, resultando así una cláusula (C^{2a}) consistente; sin embargo, estas tres tuplas positivas resultan de la expansión de una única tupla (+p3) del conjunto inicial (T_1), por lo que esta cláusula no es completa.
- Por otro lado, existe un literal (L_2^b) que también cubre a tres tuplas en el conjunto intermedio (T_2) y a ninguna negativa, por lo que sirve para construir otra cláusula (C^{2b}) consistente. Además, estas tres tuplas proceden de las tres tuplas positivas (+p1, +p2 y +p3) del conjunto inicial (T_1), por lo que la cláusula será, además, completa, y finalizará la construcción de la definición.

Usando heurísticos de evaluación que midan sus argumentos sobre los conjunto intermedios (por ejemplo, la función *Ganancia*, o la función *Interés*, tal y como veremos más adelante), no podemos prevenir que ocurran errores de este tipo, ya que los valores resultantes son iguales para ambos candidatos. Sin embargo, es posible realizar las medidas sobre el conjunto inicial (T_1), proyectando sobre él los sucesivos conjuntos intermedios (T_i, i >1); en este caso la función *Interés* (que hemos denominado *Interés** para diferenciarla), tomaría valores más acordes con el con-

junto cubierto: 1.0 para el caso de la cláusula completa, y 0.333 para la otra opción. Veremos más adelante cómo medimos el valor *Interés** a partir de la función *Interés*.

3.3 Heurísticos basados en medidas de interés

3.3.1 Definición del interés de una regla

Podemos construir un heurístico que permita evaluar los literales candidatos y prevenga los errores detectados en la función *Ganancia*. Para ello nos basaremos en medidas del interés de una regla lógica, tal y como se describe en [Piatetsky-Shapiro, 89] para la función RI (*Rule Interest*).

La función *RI*, a diferencia de la *Ganancia*, no se basa en la teoría de la información (no mide la información aportada por cada tupla positiva dentro del conjunto de entrenamiento), sino que mide el *interés* de una regla lógica de la forma A→B, en función del número de casos cubiertos por el antecedente (A) y el consecuente (B).

Para una regla A→B denominaremos:

N al tamaño (número de tuplas) del conjunto de entrenamiento,

N_A al número de hechos que satisfacen el antecedente A,

N_B al número de hechos que satisfacen el consecuente B,

$N_{A\wedge B}$ al número de hechos que satisfacen simultáneamente A y B.

Se puede llegar a la función *RI* a partir de las características que deseamos en ella, para luego construirla "a la medida". Por ejemplo, algunas condiciones que debe cumplir *RI* son:

1. Si A y B son estadísticamente independientes, la regla A→B no tendrá interés ninguno. Esto se puede expresar como:

   $$Pro(B|A) = Pro(B) \Rightarrow RI = 0$$

   y teniendo en cuenta las siguientes igualdades:

   $Pro(A\wedge B) = Pro(B|A) \cdot Pro(A)$ (probabilidad condicional)

   $Pro(A\wedge B) = N_{A\wedge B} / N$ (definición de probabilidad)

   $Pro(A) = N_A / N$

   $Pro(B) = N_B / N$

   podemos deducir la condición equivalente:

   $$N_{A\wedge B} = N_A \cdot N_B / N \Rightarrow RI = 0 \qquad \text{[EC. 3.1]}$$

2. El valor de la función RI debe crecer de forma monótona con $N_{A\wedge B}$ si los demás parámetros son constantes.
3. La función RI debe decrecer de forma monótona al crecer N_A si los demás parámetros son constantes.
4. La función RI debe decrecer de forma monótona al crecer N_B si los demás parámetros son constantes.

Se puede comprobar que la función más sencilla que cumple estas cuatro condiciones es:

$$RI_1(A \rightarrow B) = N_{A \wedge B} - N_A \cdot N_B / N \qquad \text{[EC. 3.2]}$$

que, intuitivamente, puede ser interpretada como la diferencia entre el número real de tuplas para las que se satisfacen A y B (i.e. $N_{A \wedge B}$) y el número esperado de tuplas que se satisfarían si A y B fuesen estadísticamente independientes.

Otra función que cumple las cuatro condiciones anteriores y, además, tiene la ventaja de estar normalizada (acotada en el intervalo [−1, 1]) es la expresión:

$$RI_2(A \rightarrow B) = \frac{N_{A \wedge B} - N_A \cdot N_B / N}{\sqrt{N_A \cdot N_B \cdot (1 - N_A / N) \cdot (1 - N_B / N)}} \qquad \text{[EC. 3.3]}$$

que no es más que una medida de la correlación[1] entre A y B.

Puede ampliarse el conjunto de condiciones que debe cumplir la función RI; por ejemplo, incluyendo la siguiente condición:

- El interés de la regla R debe decrecer al aumentar su complejidad. En este caso, midiendo la complejidad como la longitud de codificación en bits (*LDI(R)*, según la ecuación [EC. 2.25]), se deberá cumplir que el interés se anula cuando se supere la longitud de codificación extensional (LDE(R), aplicando la ecuación [EC. 2.22]):

  $$LDI(R) > LDE(R) \Rightarrow RI(R) = 0$$

Una posible expresión que satisface esta condición, además de las cuatro anteriores, es la siguiente:

$$RI_3(R) = RI_2(R) \cdot \max\left(\frac{LDE(R) - LDI(R)}{LDE(R)}, 0\right) \qquad \text{[EC. 3.4]}$$

El establecimiento de nuevas restricciones para la función RI (por ejemplo, añadiendo costes al uso de ciertos predicados, o limitando el número de nuevas variables en cada cláusula) deja abierta una posible línea de investigación.

La expresión que utilizaremos a partir de aquí para referirnos al *interés* de una regla será la de la ecuación [EC. 3.3].

---

1. Según el diccionario de la Real Academia Española ([RAE, 94]) la correlación se define como la "correspondencia o relación recíproca entre dos o más cosas o series de cosas; en matemáticas, es la existencia de mayor o menor dependencia mutua entre dos variables aleatorias".
El coeficiente de correlación mide la dependencia entre dos variables cuantitativas ([Larousse, 96]); su valor será positivo si las variaciones de ambas son del mismo sentido, negativo en el caso contrario y nulo si las variaciones de uno no parecen seguir en modo alguno las del otro. Se utilizan diversos coeficientes para evidenciar los vínculos que existen entre dos o más fenómenos; el más importante de ellos es el coeficiente de correlación lineal, que es tanto más próximo a +1 o a -1 cuanto más estrecho es el vínculo entre dos variables, y tanto más próximo a 0 cuanto más laxo es.
En nuestro caso, decimos que RI es una medida de la correlación porque cumple las condiciones necesarias para ello:
1) A y B independientes $\Rightarrow Pro(B|A) = Pro(B) \Rightarrow N_{A \wedge B} = N_A \cdot N_B / N \Rightarrow RI = 0$
2) A y B dependientes y varían en mismo sentido $\Rightarrow Pro(B|A) = 1 \Rightarrow N_{A \wedge B} = N_A = N_B \Rightarrow RI = 1$
3) A y B dependientes y varían en sentido contrario $\Rightarrow Pro(B|A) = 0 \Rightarrow N_{A \wedge B} = 0 \Rightarrow RI = -1$

3.3.2 Adaptación de *RI* para evaluar literales en FOIL: Interés de un literal

En principio RI (ecuación [EC. 3.3]) es aplicable directamente a reglas lógicas de la forma $A \rightarrow B$, que pueden pertenecer a la lógica de proposiciones o a la lógica de predicados de primer orden. Esta medida permite seleccionar el mejor antecedente A para un consecuente dado B.

Podemos adaptar la función RI para su uso como heurístico de evaluación de literales en FOIL. Para ello, será necesario expresar las cláusulas que se construyen en la forma $A \rightarrow B$, siendo A el literal que se desea evaluar, y B la cláusula en construcción (antes de añadir el literal candidato).

Dada una cláusula de Horn con cabeza:

$$C^i \equiv [P :- L_1, L_2, ..., L_i]$$

equivalente a la expresión lógica:

$$C^i \equiv (L_1 \wedge L_2 \wedge ... \wedge L_i \rightarrow P)$$

podemos transformarla con las siguientes equivalencias:

$$\begin{aligned}
& L_1 \wedge L_2 \wedge ... \wedge L_i \rightarrow P && \leftrightarrow \\
\leftrightarrow \quad & P \vee \neg(L_1 \wedge \neg L_2 \wedge ... \wedge \neg L_{i-1} \wedge \neg L_i) && \leftrightarrow \\
\leftrightarrow \quad & P \vee \neg L_1 \vee \neg L_2 \vee ... \vee \neg L_{i-1} \vee \neg L_i && \leftrightarrow \\
\leftrightarrow \quad & (P \vee \neg(L_1 \wedge \neg L_2 \wedge ... \wedge \neg L_{i-1})) \vee \neg L_i && \leftrightarrow \\
\leftrightarrow \quad & (L_1 \wedge L_2 \wedge ... \wedge L_{i-1} \rightarrow P) \vee \neg L_i && \leftrightarrow \\
\leftrightarrow \quad & L_i \rightarrow (L_1 \wedge L_2 \wedge ... \wedge L_{i-1} \rightarrow P)
\end{aligned}$$

De este modo, conseguimos una expresión de la forma

$$C^i \equiv A \rightarrow B$$

en la que

$$A \equiv L_i$$

$$B \equiv (L_1 \wedge L_2 \wedge ... \wedge L_{i-1} \rightarrow P) \equiv C^{i-1}$$

y el interés de $A \rightarrow B$ permitirá evaluar los literales L_i candidatos para construir la cláusula C^i.

Obsérvese que la expresión de B es la cláusula C^{i-1} (formada por los literales $L_1, L_2, ..., L_{i-1}$), construida en la iteración anterior del algoritmo de construcción de cláusulas de FOIL:

$$C^i \equiv L_i \rightarrow C^{i-1}$$

por lo que las tuplas + y − de T_i (tal como se definió en el apartado 2.5.3.2) serán, respectivamente, ejemplos y contraejemplos de B.

Estrictamente hablando, la condición lógica representada por B implica que, además de las tuplas positivas de T_i, las tuplas que no satisfacen los antecedentes de C^{i-1} también satisfarían B (serían ejemplos de B), aunque, en la práctica, no serían de utilidad para seleccionar L_i, por lo que no las tendremos en cuenta.

Del mismo modo, restringiremos las tuplas que satisfacen A a las que satisfacen L_i dentro del conjunto T_i, de modo que T_i servirá como conjunto local de entrenamiento para construir C^i a partir de C^{i-1}.

Si denominamos $T_i^{[i+1]}$ al subconjunto de tuplas de T_i que satisfacen L_i, es decir:

$$T_i^{[i+1]} = \{t \in T_i \mid \models_t(L_i)\} \qquad \text{[EC. 3.5]}$$

tal y como se indica en la figura 3-5, el valor de los argumentos de RI será el siguiente:

N	$= N_i^+ + N_i^-$	número total de tuplas de T_i
N_A	$= N_i^{[i+1]+} + N_i^{[i+1]-}$	número de tuplas (+ y −) de T_i que satisfacen L_i
N_B	$= N_i^+$	número de tuplas de T_i que satisfacen la cláusula C^{i-1}
		(aplicando [EC. 2.6] se reduce a las tuplas + de T_i)
$N_{A \wedge B}$	$= N_i^{[i+1]+}$	número de tuplas de T_i que satisfacen L_i y C^{i-1}
		(se reduce a las tuplas + de T_i que satisfacen L_i)

Aplicando la ecuación [EC. 3.3] sobre estos argumentos se obtiene la siguiente expresión para el interés de un literal L_i:

$$\text{Interés}(L_i) = = \frac{N_i^- \cdot N_i^{[i+1]+} - N_i^- \cdot N_i^{[i+1]-}}{\sqrt{N_i^- \cdot N_i^+ \cdot (N_i^{[i+1]+} + N_i^{[i+1]-}) \cdot ((N_i^+ + N_i^-) - (N_i^{[i+1]+} + N_i^{[i+1]-}))}} \qquad \text{[EC. 3.6]}$$

$N_i^+ = |T_i^+|$
$N_i^- = |T_i^-|$
$N_i^{[i+1]+} = |T_i^{[i+1]+}|$
$N_i^{[i+1]-} = |T_i^{[i+1]-}|$

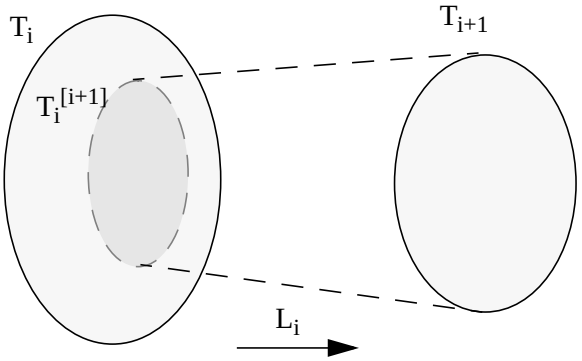

Figura 3-5: Argumentos de Interés (medidos sobre T_i)

Esta misma ecuación puede expresarse en función de los argumentos usados para el cálculo de la *Ganancia* (ecuaciones [EC. 2.29] y [EC. 2.28]):

$N_i^+ = |T_i^+|$
$N_i^- = |T_i^-|$
$N_i^{++} = |T_i^{[i+1]+}|$
$N_i^{-+} = |T_i^{[i+1]-}|$

resultando la siguiente expresión:

$$\text{Interés}(L_i) = \frac{N_i^- \cdot N_i^{++} - N_i^+ \cdot N_i^{-+}}{\sqrt{N_i^- \cdot N_i^+ \cdot (N_i^{++} + N_i^{-+}) \cdot ((N_i^+ + N_i^-) - (N_i^{++} + N_i^{-+}))}} \qquad \text{[EC. 3.7]}$$

Con esta función se solucionan algunos de los errores descritos en el apartado 3.2:

- El *Interés* depende del número de tuplas negativas cubiertas (N_i^{-+}), por ello puede corregir el error de tipo I, como se ilustra en la figura 3-2.
- El *Interés* crece con el número de tuplas positivas cubiertas, por ello corrige también el error de tipo II (figura 3-3)

Sin embargo, como sus argumentos son medidos sobre el conjunto local T_i, tampoco diferencia entre los candidatos de la figura 3-4. La solución a este último error se trata a continuación.

3.3.3 *Interés**: aplicación de *Interés* sobre conjuntos proyectados

Para solucionar el problema de tipo III es necesario definir un nuevo conjunto de tuplas para la búsqueda de L_i. Este conjunto debe ser un subconjunto de T_1 para que los argumentos que midamos en él sean representativos del conjunto cubierto por la cláusula.

Obtendremos este nuevo conjunto proyectando el conjunto intermedio T_i sobre el conjunto inicial T_1, y lo llamaremos $T_1^{[i]}$ (ver figura 3-6). Es decir, $T_1^{[i]}$ es el conjunto cubierto por la cláusula en construcción C^{i-1}. Se cumple que:

$$T_1^{[i]} = T_C(C^{i-1}) \subseteq T_1$$

Este conjunto será diferente para cada uno de los literales que se añaden a la cláusula en construcción, del mismo modo que varían los conjuntos intermedios T_i. Para el primer literal de una cláusula, coincidirá con el conjunto de entrenamiento de la cláusula (T_1):

$$T_1^{[1]} = T_C(C^0) = T_1$$

Podemos aplicar la misma función *RI* sobre estos nuevos conjuntos, razonando de modo similar a como hicimos en el apartado anterior, para evaluar los literales candidatos durante la construcción de una cláusula.

$N_i^+ = |T_i^+|$
$N_i^- = |T_i^-|$
$N_i^{[i+1]+} = |T_i^{[i+1]+}|$
$N_i^{[i+1]-} = |T_i^{[i+1]-}|$

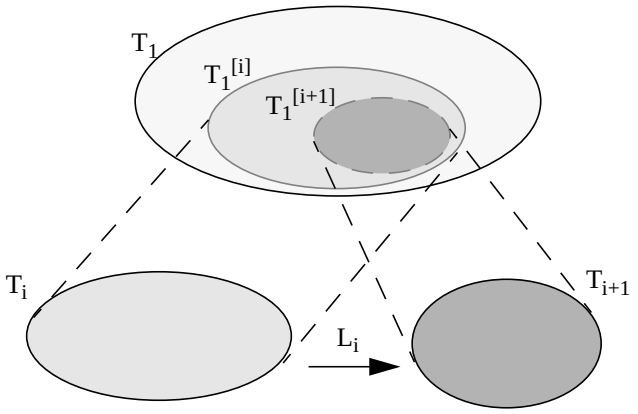

Figura 3-6: Argumentos de Interés* (medidos sobre $T_1^{[i]}$)

Para la cláusula

$$C^i \equiv A \rightarrow B$$

en la que

$$A \equiv L_i$$

$$B \equiv (L_1 \wedge L_2 \wedge ... \wedge L_{i-1} \rightarrow P) \equiv C^{i-1}$$

tendremos los siguientes argumentos, medidos en $T_1^{[i]}$:

$N = N_1^{[i]+} + N_1^{[i]-}$ número total de tuplas de $T_1^{[i]}$

$N_A = N_1^{[i+1]+} + N_1^{[i+1]-}$ número de tuplas (+ y –) de $T_1^{[i+1]}$ (satisfacen L_i)

$N_B = N_1^{[i]+}$ número de tuplas de $T_1^{[i]}$ que satisfacen cláusula C^{i-1}

$N_{A \wedge B} = N_1^{[i+1]+}$ número de tuplas de $N_1^{[i]}$ que satisfacen L_i y C^{i-1} (se reduce a las tuplas + de $T_1^{[i+1]}$)

La expresión [EC. 3.3] aplicada sobre estos argumentos queda del siguiente modo:

$$\text{Interés}^*(L_i) = \frac{N_1^{[i]-} \cdot N_1^{[i+1]+} - N_1^{[i]+} \cdot N_1^{[i+1]-}}{\sqrt{N_1^{[i]-} \cdot N_1^{[i]+} \cdot (N_1^{[i+1]+} + N_1^{[i+1]-}) \cdot ((N_1^{[i]+} + N_1^{[i]-}) - (N_1^{[i+1]+} + N_1^{[i+1]-}))}} \quad \text{[EC. 3.8]}$$

siendo

$N_1^{[i]+} = |T_1^{[i]+}|$

$N_1^{[i]-} = |T_1^{[i]-}|$

$N_1^{[i+1]+} = |T_1^{[i+1]+}|$

$N_1^{[i+1]-} = |T_1^{[i+1]-}|$

Con esta función se solucionan todos los errores descritos en el apartado 3.2:

- Por basarse en la medida del interés, se corrigen los errores I y II (apartado 3.3.2).
- Por usar el conjunto $T_1^{[i]} = T_C(C^{i-1}) \subseteq T_1$ también se corrige el error de tipo III (figura 3-4).

3.4 Ejemplos. Comparación de resultados

En muchos de los ejemplos analizados no se produce ninguno de los errores descritos para la función *Ganancia*, de modo que no existe diferencia entre las definiciones inducidas con uno u otro heurístico de evaluación. Sin embargo, en algunas ocasiones, las consecuencias de estos errores resultan bastante notables, como veremos a continuación.

3.4.1 Ejemplo de los *enfermos*

En este ejemplo, presentado por primera vez en [Gómez y Fernández, 92], podemos ver un caso extremo de cómo puede afectar el heurístico de evaluación a las definiciones inducidas. Dependiendo de qué función se emplee para evaluar los literales candidatos, se construirá una definición sencilla, más compleja, o incluso no se podrá construir ninguna válida.

Supongamos la siguiente base de datos, con información sobre diversos atributos de un conjunto de personas: estar enfermo, tener barba y ser fumador; y relaciones existentes entre ellas: ser padre de alguien o ser jefe de alguien. Los atributos se representan por relaciones unarias en la base de datos, mientras que las relaciones entre personas son binarias:

Tabla 3-1: enfermo

b3	b4	c3	c4	c5
d2	d3	d4		

Tabla 3-2: barbudo

a3	a4	b4	c1	c2
c4	d1	d2		

Tabla 3-3: fumador

b3	b4	d2		

Tabla 3-4: padre_de

a1, b1	a1, b2	a2, b1	a2, b2	a3, b3
a3, b4	a4, b3	a4, b4	b1, c1	b1, c2
b2, c3	b3, c3	b4, c4	b4, c5	c1, d1
c2, d1	c2, d2	c2, d3	c4, d2	c4, d3
c5, d4				

Tabla 3-5: jefe_de

a1,b1	a2,b2	a2,b3	a3,b2	a3,b3
a4,b3	b1,c1	b2,c2	b2,c3	b2,c4
b3,c2	b3,c3	b3,c4	b4,c4	b4,c5
c1,d1	c2,d1	c3,d2	c3,d3	c3,d4
c4,d2	c4,d3	c4,d4	c5,d4	

En las siguientes figuras se representa esta misma información de forma gráfica. Para representar a las personas “enfermas” se ha sombreado su cara; los atributos “fumador” y “barbudo” se indican con una pipa y una barba de chivo respectivamente. Las relaciones binarias se indican mediante líneas verticales en la figura 3-7 (“padre_de”) y figura 3-8 (“jefe_de”); que deben leerse

de arriba a abajo, para interpretar correctamente la semántica de “a es padre de b” o “a es jefe de b” respectivamente.

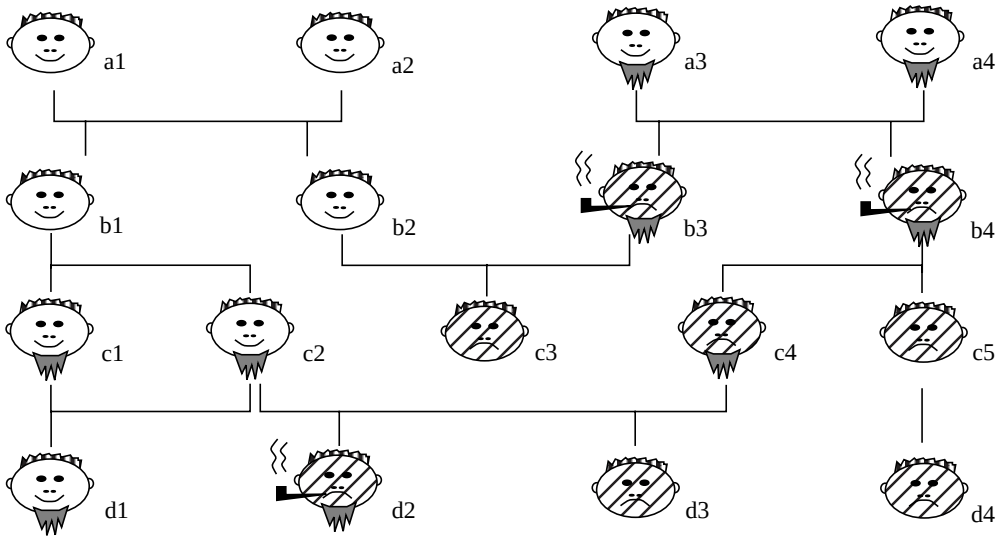

Figura 3-7: Relación “padre_de”

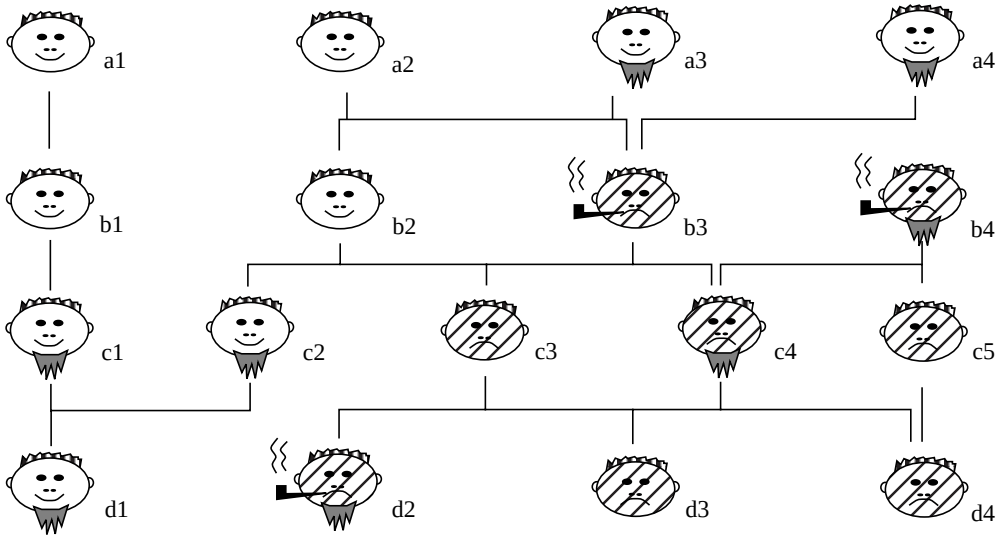

Figura 3-8: Relación “jefe_de”

Supongamos que queremos encontrar una definición lógica para la relación *enfermo*, usando relaciones existentes en la base de datos (puede intentarse este ejercicio manualmente, inspeccionando las figuras figura 3-7 y figura 3-8). Usando FOIL, se introducirían las relaciones definidas anteriormente como dato de entrada, indicando que *enfermo* debe ser el predicado objetivo. Las tuplas + serán por tanto las personas enfermas, y las negativas las no enfermas:

T_1 = {+<b3>, +<b4>, +<c3>, +<c4>, +<c5>, +<d2>, +<d3>, +<d4>, –<a1>, –<a2>,

–<a3>, –<a4>, –<b1>, –<b2>, –<c1>, –<c2>, –<d1>}

Según utilicemos como heurístico para evaluar los literales la función *Ganancia*, la función *RI* o la función *RI**, el sistema FOIL inducirá una definición diferente.

Si estudiamos en detalle las definiciones construidas, observamos que ya en el primer literal de la primera cláusula se produce la primera diferencia, ya que con la función *Ganancia* el literal elegido es *jefe_de(V_2, X_1)*, mientras que con las funciones *RI* y *RI** es *padre_de(V_2, X_1)*, según puede verse en la figura 3-9.

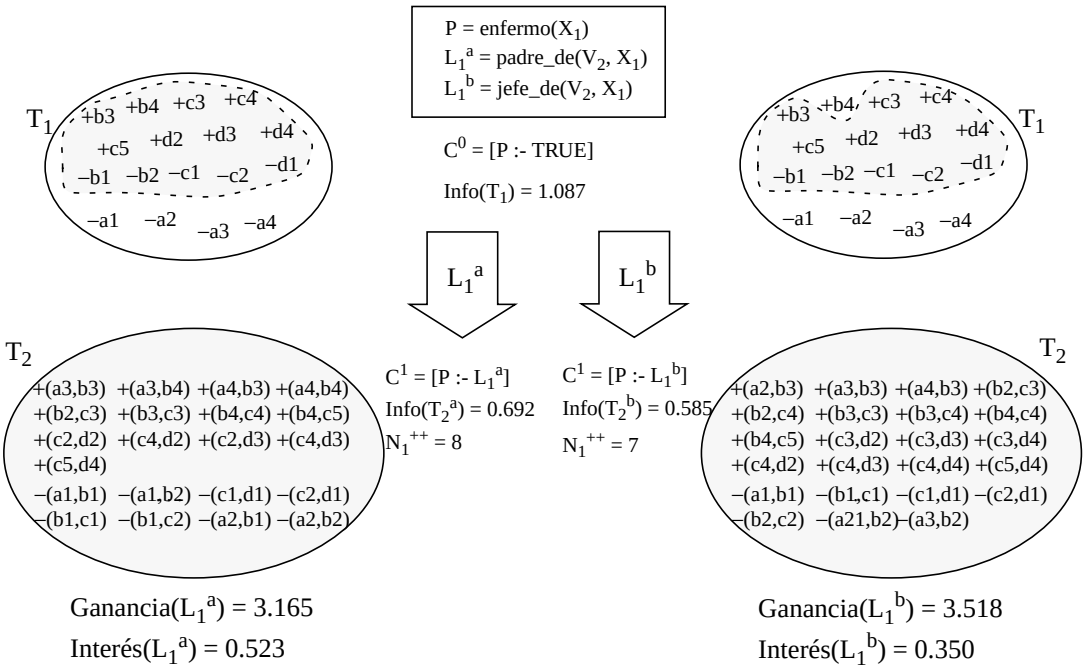

Figura 3-9: Candidatos para primer antecedente de primera cláusula

Como puede observarse, la diferencia en este caso se debe a un error de tipo II, pues el literal *jefe_de(V_2, X_1)* cubre menos tuplas + que *padre_de(V_2, X_1)* y, sin embargo, produce un conjunto T_2 con mayor concentración de tuplas + (es decir, con menor *información*).

En este ejemplo, la función *Ganancia* favorece erróneamente a *jefe_de(V_2, X_1)*, mientras que las funciones *RI* y *RI** permiten elegir al mejor literal *padre_de(V_2, X_1)*.

Si continuamos la ejecución utilizando las funciones *RI* y *RI**, encontramos una nueva diferencia en la búsqueda del siguiente literal (figura 3-10). Esta diferencia se debe a un error de tipo III, por el cual el literal *barbudo(V_2)*, que cubre más tuplas en el conjunto T_2, se ve favorecido por *RI* frente al literal *enfermo(V_2)*, a pesar de que este último cubra más tuplas del conjunto inicial T_1. Con la función *RI** se previene este error y se selecciona correctamente el literal *enfermo(V_2)*.

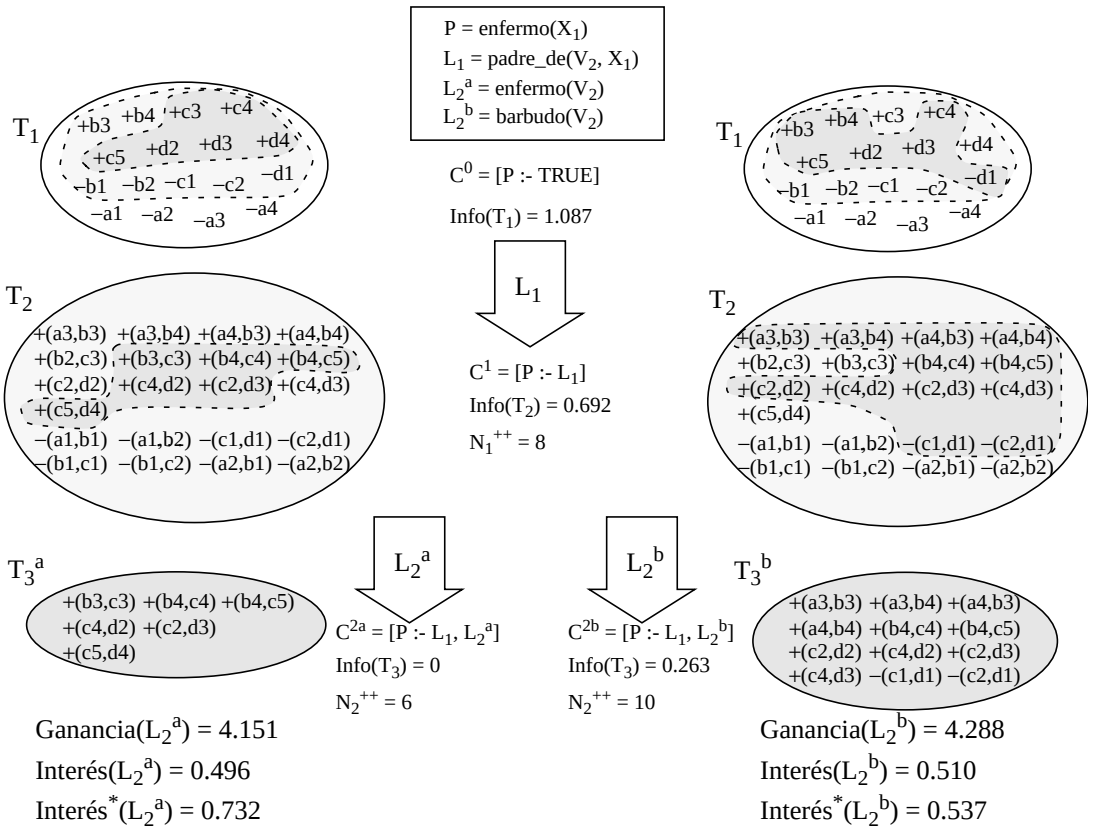

Figura 3-10: Candidatos para segundo antecedente de primera cláusula

Para finalizar esta discusión, vamos a comparar los resultados que se obtienen en este ejemplo usando los diferentes heurísticos de evaluación de literales:

- $D_{Ganancia}$[1] =

  enfermo(X_1) :- jefe_de(V_2, X_1), enfermo(V_2), ¬fumador(V_2)

  enfermo(X_1) :- fumador(X_1)

  enfermo(X_1) :- jefe_de(V_2, X_1), enfermo(V_2), padre_de(V_2, X_1)

- $D_{Interés}$ =

  enfermo(X_1) :- padre_de(V_2, X_1), barbudo(V_2), fumador(X_1)

  enfermo(X_1) :- padre_de(V_2, X_1), enfermo(V_2)

- $D_{Interés*}$ =

1. Realmente el principio de Mínima Longitud de Descripción no permitiría la construcción de una definición tan compleja como ésta, sino que la podaría eliminando su última cláusula, con lo que resultaría una definición incompleta:

   $D_{Ganancia}$ =

   enfermo(X_1) :- jefe_de(V_2, X_1), enfermo(V_2), ¬fumador(V_2)
   enfermo(X_1) :- fumador(X_1)

enfermo(X_1) :- padre_de(V_2, X_1), enfermo(V_2)

enfermo(X_1) :- fumador(X_1)

Como puede verse a simple vista, la definición $D_{Ganancia}$ es la peor de las tres, ya que es la más compleja; de hecho, debería finalizar su construcción sin incluir su última cláusula, tal y como se ha indicado, resultando así una definición incompleta. La definición $D_{Interés}$ es más sencilla, lo que permite que se construya cumpliendo las condiciones de completitud y consistencia. Sin embargo, la mejor definición es la que se obtiene con el heurístico *Interés**: "los enfermos son hijos de enfermos o son fumadores".

Si consideramos que la complejidad de una definición se puede medir por el número de bits necesarios para codificarla (apartado 2.5.2.9), tendremos los siguientes valores:

$LDI(D_{Ganancia}) = 13.51 + 4.17 + 15.51 - \log_2(3!) = 30.61$ bits

$LDI(D_{Interés}) = 13.51 + 9.92 - \log_2(2!) = 22.43$ bits

$LDI(D_{Interés*}) = 9.92 + 4.17 - \log_2(2!) = 13.10$ bits

Por tanto, con la función Interés* no sólo se produce una importante simplificación (ahorro de bits), sino que, en ocasiones, permite obtener una definición consistente y completa que con otras funciones de evaluación (*Ganancia* en nuestro ejemplo) no se pueden construir.

3.4.2 Ejemplo de los *abuelos*

Este es otro sencillo ejemplo en el que los resultados obtenidos por el algoritmo FOIL dependen del heurístico de evaluación que se emplee al construir las reglas. La base de datos de entrada consta de cuatro relaciones:

abuelo(A) = la persona A es un abuelo

padre_de(A, B) = la persona A es padre de B

varon(A) = la persona A es un varón

tiene_hijos(A) = la persona A tiene hijos

Los atributos usados en las tuplas de estas relaciones pertenecen al siguiente dominio:

personas: {a1, a2, a3, b1, b2, b3, b4, b5, b6, c1, c2, c3}

y las tuplas que forman cada relación son las siguientes:

Tabla 3-6: abuelo

a1	a2			

Tabla 3-7: padre_de

a1, b4	a1, b5	a2, b1	a2, b2	a2, b3
a2, b5	a3, b6			

Tabla 3-8: varón

b1	b2	b3		

Tabla 3-9: tiene_hijos

a1	a2	a3	b5	

Podemos representar estas relaciones del siguiente modo:

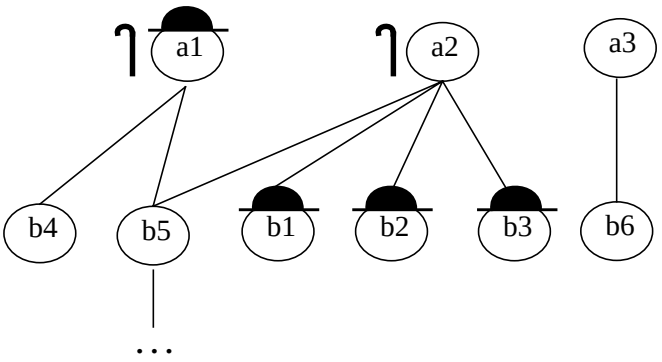

Figura 3-11: Ejemplo de los abuelos

La relación objetivo será *abuelo*. Hay que hacer notar que en este ejemplo tan pequeño, la longitud de descripción extensional asociada a la relación objetivo es muy pequeña (ecuación [EC. 2.21]):

$$\mathrm{LDE}(\textit{abuelo}) = \log_2(12) + \log_2(\binom{12}{2}) = 9.63 \text{ bits}$$

por tanto, sólo serán admisibles descripciones cortas, que no superen esta longitud. Esto hará que sólo con el heurístico *Interés** se consiga una definición válida, además de consistente y completa.

Las definiciones construidas con las diferentes funciones de evaluación son las siguientes:

- $D_{Ganancia}$ =

  abuelo(X_1) :- padre_de(X_1, V_2), varon(V_2)

  abuelo(X_1) :- padre_de(X_1, V_2), ...

  al aplicar poda por Mínima Longitud de Descripción, finaliza en este punto la construcción de la última cláusula que, además, es eliminada porque su precisión (ver nota de pág. 73) es inferior al umbral permitido:

  precisión = 2/3 = 67 % < 85%

  Por tanto, resultaría una definición incompleta (no cubre la tupla +<a1>).

Sin embargo, teniendo en cuenta que el principio de Mínima Longitud de Descripción debe aplicarse para comparar el conjunto cubierto (formado por una única tupla: +<a2>) con la definición intensional, resulta que no se satisface para esta definición:

$LDE(T_C(D_{Ganancia})) = \log_2(12) + \log_2(\binom{12}{1}) = 7.170$ bits

$LDI(D_{Ganancia}) = 9.585$ bits

con lo que no se obtiene ninguna definición intensional válida para la relación *abuelo*.

- $D_{Interés}$ =

  abuelo(X_1) :- padre_de(X_1, V_2), varon(V_2)

  abuelo(X_1) :- padre_de(X_1, V_2), ...

  el proceso resultante es el mismo que en el caso de la función *Ganancia*, de modo que la definición construida no es válida, y no se obtiene ninguna definición intensional para *abuelo*.

- $D_{Interés*}$ =

  abuelo(X_1) :- padre_de(X_1, V_2), tiene_hijos(V_2)

  En este caso sí se satisface el principio de Mínima Longitud de Descripción:

  $LDE(T_C(D_{Interés*})) = \log_2(12) + \log_2(\binom{12}{2}) = 9.629$ bits

  $LDI(D_{Interés*}) = 9.585$ bits

  Obtenemos de este modo una definición consistente y completa con el heurístico Interés*.

Capítulo 4:

INDUCCIÓN DE CONOCIMIENTO BORROSO DE PRIMER ORDEN[1]

En este capítulo trataremos de generalizar todos los conceptos relacionados con la lógica de predicados de primer orden, para poder aplicarlos en el caso de lógica borrosa. Como veremos, las principales diferencias se producen en la semántica asociada en uno u otro caso.

A continuación se describirá el sistema FZFOIL, construido a partir de FOIL, para inducir conocimiento con incertidumbre a partir de bases de datos relacionales borrosas.

4.1 Lógica borrosa de primer orden

Aunque existe gran cantidad de trabajos enfocados a formalizar la lógica borrosa, desde un punto de vista matemático, la mayoría de ellos están orientados hacia la lógica de proposiciones o la lógica de proposiciones extendida; por ejemplo, [Fernández y Sáez-Vacas, 95] constituye una buena referencia. Sin embargo, no hemos encontrado ningún trabajo en el que se trate la lógica borrosa de predicados, más allá de la simple definición de relaciones borrosas (apartado A.3).

¿Puede pensarse que, conocida la lógica de primer orden y la lógica borrosa de proposiciones, se conoce la lógica borrosa de predicados? Sin duda, algunas de las definiciones (como las referentes a la sintaxis o algunas referentes a la semántica como la asignación de variables) no varían al pasar de la lógica clásica de predicados a la lógica borrosa de predicados, pero otras (satisfacción, sentencias condicionales, etc.) son diferentes y, aunque son una extensión de la versión booleana, conviene definirlas de antemano, ya que podrían existir otras extensiones válidas.

1. Hay que observar que, cuando se habla de “conocimiento de primer orden”, en realidad se está utilizando una elipsis de “conocimiento *modelado con lógica* de primer orden”. Lo mismo ocurre con “conocimiento borroso de primer orden”.

4.1.1 Sintaxis

La sintaxis que utilizaremos en lógica borrosa de primer orden es la misma que se utiliza en la lógica clásica de predicados de primer orden, y que puede verse, por ejemplo, en [Fernández y Sáez-Vacas, 95]. Se utilizarán las mismas definiciones para términos, fórmulas atómicas, secuencias de formación, sentencias, cuantificadores, etc.

4.1.2 Semántica

Desde el punto de vista de la semántica es donde se producen los mayores cambios al pasar de lógica booleana a lógica borrosa. Por ello redefiniremos los principales conceptos relacionados con la semántica, generalizándolos a partir de su definición para lógica booleana (pueden verse también en [Fernández y Sáez-Vacas, 95]).

4.1.2.1 Conceptualización

En lógica borrosa de predicados se define una *conceptualización* como un modelo conceptual que consta de:

- Un *universo del discurso*, U, conjunto (que puede ser infinito) de individuos u objetos a considerar.
- Un conjunto finito (que puede ser vacío), R, de *relaciones borrosas* entre elementos de U.

  Una *relación borrosa* $\underline{P}$ de grado n es un subconjunto borroso del producto cartesiano U^n (apartado A.3), es decir, un subconjunto borroso de tuplas:

  $$\{<(e_{i1}, e_{i2},..., e_{in}), \mu_P(e_{i1}, e_{i2},..., e_{in})>\}, \text{ donde } e_{ij} \in U.$$

  Una *propiedad borrosa* es una relación borrosa de grado 1. En el conjunto R pueden existir también relaciones ordinarias, pues éstas se consideran un caso particular de relación borrosa.
- Un conjunto finito (que puede ser vacío), F, de *funciones borrosas*[1], que hacen corresponder a ciertos elementos de U otros elementos de U.

4.1.2.2 Interpretación

Una interpretación i, en lógica borrosa, se define del mismo modo que en lógica booleana: es una función que aplica elementos del lenguaje en elementos de la conceptualización, y que debe satisfacer las siguientes condiciones:

- Si c es un símbolo de constante, entonces $i(c) \in U$ (es decir, las constantes representan individuos del universo del discurso).

1. En nuestro desarrollo no utilizaremos el concepto de *función borrosa*, por ello evitaremos su definición en este punto. En caso necesario, podría extenderse la definición de función de la lógica de predicados (aplicación $U^n \rightarrow U$), para su versión borrosa en lógica borrosa de predicados.
De hecho, una de las limitaciones actuales de FOIL y de FZFOIL, es que no utilizan funciones en las definiciones lógicas que inducen. La utilización de funciones borrosas en FZFOIL se propone en el apartado 6.2.2 como posible futura mejora, para ampliar la capacidad expresiva del lenguaje de representación del conocimiento.

- Si $\underline{P}$ es un símbolo de predicado borroso de grado n (es decir, que se aplica a n términos para formar una fórmula atómica), entonces $i(\underline{P}) \subset U^n$ (es decir, los símbolos de predicado representan relaciones borrosas de la conceptualización).
- Si f es un símbolo de función borrosa de grado n, entonces i(f) representa una función borrosa de la conceptualización (independientemente de cómo se definan las funciones borrosas).

Por tanto, el concepto de interpretación no es más que una formalización de la noción de que los símbolos *representan* aspectos de la realidad modelada en la conceptualización. Y hacer una conceptualización no es más que *acotar la realidad*, es decir, listar todos los individuos u objetos a considerar y todas las relaciones entre ellos.

4.1.2.3 Asignación de variables

Una *asignación de variables*, A, es una función que hace corresponder elementos de U a las variables que figuran en las sentencias. Las constantes corresponden a los elementos de U según define la interpretación, y las funciones se aplican también a elementos de U, según su interpretación. En general, dada una interpretación i y una asignación de variables A, podemos extender A a una *asignación de términos*:

- Si c es un símbolo de constante, A(c) = i(c);
- Si f es un símbolo de función,

$$A(f(t_1, t_2,..., t_n)) = F(x_1, x_2,..., x_n), \text{ donde } F = i(f) \text{ y } x_i = A(t_i)$$

4.1.2.4 Conjunto de valores de verdad

El *conjunto de valores de verdad* en lógica borrosa lo definimos como el conjunto de números reales en el intervalo unidad: V = [0, 1].

4.1.2.5 Evaluación

Definimos la *evaluación* de una sentencia S, para una asignación de variables A y una interpretación i, y la representamos como E(S, A, i), a una función que asigna un valor del conjunto de valores de verdad a dicha sentencia.

$$E(S, A, i) \in [0, 1] \qquad \text{[EC. 4.1]}$$

Obsérvese que pueden darse diferentes valores de evaluación para una misma sentencia, dependiendo de cual sea la asignación de variables y la interpretación que se realice.

4.1.2.6 Satisfacción

En una conceptualización, diremos que una interpretación conjuntamente con una asignación de variables, iA, *satisface en grado* σ *a una sentencia* si el valor de evaluación de la sentencia para esa iA es σ. También diremos que la sentencia *se satisface en grado* σ *para esa iA*.

La notación para expresar que la sentencia S se satisface en grado σ para iA es: $\models_{iA}(S, \sigma)$

$$\models_{iA}(S, \sigma) \quad \Leftrightarrow E(S, A, i) = \sigma \qquad \text{[EC. 4.2]}$$

Si para cada una de las reglas sintácticas de la lógica borrosa de predicados de primer orden establecemos las condiciones bajo las cuales la sentencia se satisface, estas condiciones definen la semántica de la lógica borrosa de primer orden. Son las siguientes:

- Si la sentencia es una fórmula atómica, $P(t_1, t_2,..., t_n)$, entonces

$$\models_{iA}(P(t_1, t_2,..., t_n), \sigma) \Leftrightarrow \mu_{i(P)}(A(t_1), A(t_2),..., A(t_n)) = \sigma \qquad \text{[EC. 4.3]}$$

  (una fórmula atómica se satisface en grado σ si y sólo si las asignaciones de sus argumentos resultan en una tupla incluida con un grado de verdad σ en la interpretación del predicado).

- Si la sentencia es la negación de otra, entonces

$$\models_{iA}(\neg S, \sigma) \quad \Leftrightarrow \models_{iA}(S, 1-\sigma) \qquad \text{[EC. 4.4]}$$

  (la negación de una sentencia se satisface en grado σ si y sólo si la sentencia sin negar se satisface en grado $1-\sigma$).

- Si la sentencia es la conjunción de otras dos, entonces

$$\models_{iA}(S1 \wedge S2, \sigma) \quad \Leftrightarrow \sigma = \min(\sigma_1, \sigma_2) \qquad \text{[EC. 4.5]}$$

  siendo:

$$\models_{iA}(S1, \sigma_1) \text{ y } \models_{iA}(S2, \sigma_2)$$

  (la conjunción de dos sentencias se satisface en grado σ si y sólo si el menor de los grados de satisfacción de cada una de ellas es σ; esta condición puede generalizarse a la conjunción de un número cualquiera de sentencias).

- Si la sentencia es la disyunción de otras dos, entonces

$$\models_{iA}(S1 \vee S2, \sigma) \quad \Leftrightarrow \sigma = \max(\sigma_1, \sigma_2) \qquad \text{[EC. 4.6]}$$

  siendo:

$$\models_{iA}(S1, \sigma_1) \text{ y } \models_{iA}(S2, \sigma_2)$$

  (la disyunción de dos sentencias se satisface en grado σ si y sólo si el mayor de los grados de satisfacción de cada una de ellas es σ; esta condición se puede generalizar a la disyunción de un número cualquiera de sentencias).

- Si la sentencia es un condicional, entonces

$$\models_{iA}(S1 \rightarrow S2, \sigma) \quad \Leftrightarrow \models_{iA}(\neg S1 \vee (S1 \wedge S2), \sigma) \qquad \text{[EC. 4.7]}$$

  desarrollando esta expresión resulta el siguiente valor para el grado de satisfacción:

$$\sigma = \max(1-\sigma_1, \min(\sigma_1, \sigma_2))$$

  siendo:

$$\models_{iA}(S1, \sigma_1) \text{ y } \models_{iA}(S2, \sigma_2)$$

- Si la sentencia es un bicondicional, entonces

$$\models_{iA}(S1 \leftrightarrow S2, \sigma) \quad \Leftrightarrow \sigma = \min(\sigma_D, \sigma_I) \qquad \text{[EC. 4.8]}$$

  siendo:

$$\models_{iA}(S1 \rightarrow S2, \sigma_D) \text{ y } \models_{iA}(S2 \rightarrow S1, \sigma_I)$$

  (una sentencia bicondicional se satisface en grado σ si y sólo si el menor de los grados de satisfacción de los dos condicionales en que puede descomponerse es σ).

- Si la sentencia utiliza el cuantificador universal (∀), entonces

$$\models_{iA}((\forall x)\,(S), \sigma) \quad \Leftrightarrow \text{para todo } c \in U,\ \models_{iA}(S, \sigma) \qquad \text{[EC. 4.9]}$$

con:

$$A'(x) = c,\ A'(y) = A(y) \text{ para } y \neq x$$

(una sentencia cuantificada universalmente en una variable se satisface en grado σ si y sólo si la sentencia que está dentro del alcance del cuantificador se satisface en grado σ para todas las asignaciones de la variable cuantificada).

- Si la sentencia utiliza el cuantificador existencial (∃), entonces

$$\models_{iA}((\exists x)\,(S), \sigma) \quad \Leftrightarrow \text{para algún } c \in U,\ \models_{iA}(S, \sigma) \qquad \text{[EC. 4.10]}$$

con:

$$A'(x) = c,\ A'(y) = A(y) \text{ para } y \neq x$$

(una sentencia cuantificada existencialmente se satisface en grado σ si y sólo si la sentencia que está dentro del alcance del cuantificador se satisface en grado σ para una o más de las asignaciones de la variable cuantificada).

4.1.2.7 Calificadores borrosos de verdad

En lógica borrosa de predicados definimos un *calificador borroso de verdad* V como una función que modifica el grado de verdad de una sentencia borrosa S (por analogía a la definición de calificador borroso de verdad en lógica borrosa de proposiciones, apartado A.4.2.1).

La forma de una sentencia S' con calificador borroso de verdad es la siguiente:

$$S': S \text{ es } V$$

El calificador borroso V se representa por una etiqueta lingüística ("verdadero", "muy verdadero", "algo verdadero", "muy falso", etc.) que se aplica sobre los valores de verdad de la sentencia borrosa S. Las funciones asociadas a estos calificadores pueden verse en el apéndice A, figura A-1.

El grado de satisfacción de la sentencia S', resultante de aplicar el calificador borroso de verdad V sobre una sentencia borrosa S, se calcula como el resultado de aplicar una función V, que representa al calificador, sobre el grado de satisfacción de S:

$$\models_{iA}(S', \sigma_V) \quad \Leftrightarrow \sigma_V = V(\sigma) \qquad \text{[EC. 4.11]}$$

siendo:

$$S': S \text{ es } V$$
$$\models_{iA}(S, \sigma)$$

4.1.2.8 Calificadores borrosos de probabilidad

En lógica borrosa de predicados definimos un *calificador borroso de probabilidad* P como una función que describe una restricción en la distribución de probabilidad asociada a una sentencia S (para la lógica borrosa de proposiciones puede verse la definición correspondiente en el apartado A.4.2.2).

La forma de una sentencia S' con calificador borroso de probabilidad es la siguiente:

S': Pro(S) es P

siendo Pro(S) una probabilidad asociada a la sentencia S, y P un calificador borroso de probabilidad.

Los posibles valores de P se representan con etiquetas de la forma: "probable", "muy probable", "improbable", etc. cuyo significado viene dado por curvas como las representadas en el apéndice A, figura A-3

El grado de satisfacción para la sentencia S' con calificador borroso de probabilidad será:

$$\models_{iA}(S', \sigma_P) \qquad \Leftrightarrow \sigma_P = P(Pro(iA(S))) \qquad \text{[EC. 4.12]}$$

siendo:

S': Pro(S) es P

P(Pro(iA(S))) la probabilidad asociada a S con la interpretación i y la asignación de variables A

4.2 Inducción de conocimiento con incertidumbre

Muchos sistemas de aprendizaje tradicionales son capaces de inducir conocimiento incierto: ID3 ([Quinlan, 86]), C4.5 ([Quinlan, 93]), etc., dado que asignan un factor de incertidumbre, en forma de probabilidad, al conocimiento inducido. Este tipo de representación de la incertidumbre en el conocimiento inducido también es aplicable a sistemas de aprendizaje de ILP: FOIL ([Quinlan, 90]), FOCL ([Pazzani et al., 91]), mFOIL ([Dzeroski, 91]), etc. En general, esta probabilidad puede calcularse a partir del conjunto de entrenamiento o de un conjunto de ejemplos de prueba, una vez finalizado el proceso de inducción.

Por otro lado, las reglas con factor de incertidumbre en su antecedente se pueden obtener por métodos clásicos si el factor de incertidumbre de cada atributo de entrada se expresa como un atributo más.

Respecto a la borrosidad, también existen sistemas que generan reglas borrosas: [Wang y Mendel, 92], [González y Pérez, 95], [Yeh y Chen, 96], etc. aunque su aplicación se restringe a la lógica de proposiciones extendida. En estos sistemas se permite asignar valores borrosos a los atributos utilizados en las reglas inducidas.

Sin embargo, no existen sistemas que combinen la ILP con la lógica borrosa, para inducir reglas construidas mediante literales borrosos (predicados borrosos aplicados sobre variables). En una relación borrosa, la incertidumbre no se refiere necesariamente al valor de los atributos utilizados, sino, en general, a la pertenencia de sus tuplas a la misma (apartado A.3).

Por tanto, los sistemas existentes para "inducción de reglas borrosas" sólo son capaces de manipular un caso concreto de borrosidad (la que se refiere al valor de los atributos), y no relaciones borrosas generales, en las que la borrosidad va asociada a las tuplas y no a los atributos.

Con el sistema FZFOIL, que se describe a continuación, se permitirá utilizar como dato de entrada relaciones borrosas, en las que sus tuplas tengan asignado un grado de pertenencia, así como relaciones ordinarias. Por otro lado, las reglas que se induzcan serán definiciones borrosas, tanto de relaciones ordinarias como borrosas, construidas mediante literales borrosos u

ordinarios. Estas reglas tendrán, además una medida de incertidumbre en forma de probabilidad, que indicará su precisión, útil para poder ser utilizada en la base de conocimiento de un sistema experto, por ejemplo.

4.3 FZFOIL

4.3.1 Idea básica

El propósito de FZFOIL (*FuZzy First Order Inductive Learner*) es obtener una definición lógica de una relación P (fijada por el usuario), en función de otras relaciones Q_i de la base de datos, de ella misma (permitiendo definiciones recursivas), e incluso de relaciones nuevas (inventadas por el sistema). Las relaciones de la base de datos pueden ser borrosas, y pueden venir definidas de forma extensional (por un conjunto de tuplas) o intensional (por una definición lógica).

Por tanto, se ha ampliado la potencia expresiva del sistema FOIL original ([Quinlan, 90]), tanto en la entrada de datos (datos borrosos, relaciones intensionales) como en la salida (definiciones con literales borrosos y con literales inventados).

Otra aportación interesante de FZFOIL es el empleo de la función *Interés* para evaluar literales, sustituyendo a la función Ganancia de FOIL, para corregir las deficiencias encontradas en esta última, explicadas en el apartado 3.2. Además se han modificado los conjuntos locales de entrenamiento, empleando ahora proyecciones de los mismos sobre el conjunto inicial, para poder evaluar los literales candidatos con mayor precisión y obtener a la vez una medida de la calidad de la cláusula en construcción.

Las definiciones construidas están formadas por una cláusula de Horn o por la disyunción de varias de ellas. La forma general de cada una de estas cláusulas es la siguiente:

$$p(X_1,..., X_k) :- L_1, L_2,..., L_n$$

donde la cabeza de cada cláusula es siempre el predicado objetivo p con todos sus términos variables ($X_1,..., X_k$), y el cuerpo lo forman una serie de literales ($L_1, L_2,..., L_n$). Estos literales son siempre:

a) Predicados correspondientes a alguna de las relaciones Q_i definidas inicialmente, construidos sobre términos variables.

b) O bien algunos predicados predefinidos (como el predicado igualdad "=" y el de comparación ">"), aplicado a dos variables de la cabeza, o a una variable y un término constante.

c) O bien predicados borrosos inventados a partir de otros predicados borrosos (añadiendo etiquetas como "muy", "bastante", etc.).

Los literales pueden llevar el signo "¬" de la negación lógica.

4.3.2 Algunas definiciones "borrosas"

A continuación se definen diferentes conceptos relacionados con la lógica borrosa. No han sido mencionados en el apartado 4.1 porque no son conceptos tan genéricos como los que definen la sintaxis o la semántica de la lógica borrosa de primer orden, sino que están más orientados a la inducción de conocimiento dentro de FZFOIL: literales borrosos, signo borroso, grado de cobertura, etc.

4.3.2.1 Predicados y literales borrosos

El nombre de una relación borrosa Q se denomina *predicado borroso* q, o dicho de otro modo, la interpretación del predicado borroso q es la relación borrosa Q.

Un *literal borroso* L es una fórmula atómica, formada por un predicado borroso q aplicado sobre términos variables:

$$L = q(V_1,..., V_n)$$

Un literal borroso puede llevar el símbolo "¬" de la negación lógica.

Decimos que una cláusula de Horn () es borrosa cuando está construida con literales borrosos, bien en su cabeza (consecuente) o en su cuerpo (antecedentes) o en ambos.

4.3.2.2 Signo borroso de una tupla

Al definir relaciones ordinarias, se puede hablar del signo positivo o negativo de una tupla (apartado 2.5.2.1), asociándolo a su pertenencia o no a la relación objetivo. Por el contrario, con relaciones borrosas, el concepto de signo positivo o negativo de una tupla pierde su significado original, ya que el conjunto de pertenencia (apartado A.1.1), deja de ser el par de valores {0, 1} para convertirse en el intervalo [0, 1].

Del mismo modo que existe un grado de pertenencia borroso a una relación borrosa, será necesario también borrosificar el concepto de signo de una tupla, para indicar su grado de pertenencia a la relación objetivo (borrosa).

Por similitud al caso booleano, definimos el *signo borroso* de una tupla $t = <a_1,..., a_k>$, y lo representamos como SB(t), como el grado con que t satisface (apartado 4.3.2.3) al predicado objetivo p con las variables $(V_1,..., V_k)$ de la cabeza en una cláusula de Horn. También puede definirse como el grado de pertenencia de t a la relación objetivo P. Por tanto, el signo borroso tomará valores en el intervalo [0, 1]:

$$\models_t(p(V_1,..., V_k), SB(t)) \quad \Leftrightarrow SB(t) = \mu_P(t) \in [0, 1] \qquad \text{[EC. 4.13]}$$

Si t es una tupla de una relación borrosa P, definimos el signo borroso de una tupla t', extensión suya (apartado 2.5.2.2) como el signo borroso de t:

$$SB(t') = SB(t), \quad \forall\ t' \text{ extensión de } t \qquad \text{[EC. 4.14]}$$

Podemos definir un subconjunto de tuplas positivas (con signo "+") para una relación borrosa con las tuplas cuyo signo borroso sea exactamente igual a 1:

$$T^+ = \{<t, 1> \mid SB(t) = 1\} \qquad \text{[EC. 4.15]}$$

del mismo modo, las tuplas negativas (con signo "–") serán aquéllas cuyo signo borroso sea exactamente igual a 0 (al igual que en el caso booleano, estas tuplas pueden venir explícitamente declaradas u obtenerse por la suposición de mundo cerrado):

$$T^- = \{<t, 1> \mid SB(t) = 0\} \qquad \text{[EC. 4.16]}$$

además, existirán tuplas que no sean positivas ni negativas.

$$T^{\sim} = \{<t, 1> \mid 0 < SB(t) < 1\} \qquad \text{[EC. 4.17]}$$

El conjunto de entrenamiento T para inducir una definición de una relación borrosa será un conjunto ordinario (apartado 4.3.3.4), formado por los subconjuntos de tuplas definidos anteriormente: T^+, T^- y $T^{\sim}$. Estos subconjuntos definen una partición de T:

$$T = T^+ \cup T^- \cup T^{\sim} \quad \text{[EC. 4.18]}$$

$$\emptyset = T^+ \cap T^- \cap T^{\sim} \quad \text{[EC. 4.19]}$$

4.3.2.3 Grado de satisfacción de una tupla

Decimos que una tupla t *satisface en grado* σ *un literal borroso* $L = q(V_1,..., V_n)$, o que L se satisface en grado σ para t, y lo expresamos como $\models_t(L, \sigma)$, cuando al asignar a las variables $(V_1,..., V_n)$ de L los valores de la tupla t obtenemos una nueva tupla t' cuyo grado de pertenencia a $\underline{Q}$ es σ:

$$\models_t(L, \sigma) \quad \Leftrightarrow \mu_{\underline{Q}}(t) = \sigma \quad \text{[EC. 4.20]}$$

El grado en que una tupla satisface a los antecedentes de una cláusula $(L_1 \wedge ... \wedge L_n)$ lo definimos como el mínimo de los grados de satisfacción para cada uno de ellos:

$$\models_t(L_1 \wedge L_2 \wedge ... \wedge L_n, \sigma) \Leftrightarrow \sigma = \min\{\sigma_i\}, \text{ para } i \in \{1,..., n\} \quad \text{[EC. 4.21]}$$

siendo

$$\models_t(L_i, \sigma_i)$$

Decimos que una tupla t *satisface en grado* σ *una cláusula* $C = [L_0 :- L_1,..., L_n]$, o que C se satisface en grado σ para t, y lo expresamos como $\models_t(C, \sigma)$, cuando t satisface en grado σ la sentencia: $\neg(L_1 \wedge ... \wedge L_n) \vee (L_1 \wedge ... \wedge L_n \wedge L_0)$

$$\models_t (C, \sigma) \quad \Leftrightarrow \models_t (\neg(L_1 \wedge ... \wedge L_n) \vee (L_1 \wedge ... \wedge L_n \wedge L_0), \sigma) \quad \text{[EC. 4.22]}$$

Obsérvese que estas definiciones son coherentes con las dadas en el apartado 4.1.

4.3.2.4 Grado de cobertura y conjunto cubierto

Una cláusula de Horn $C = [L_0 :- L_1 \wedge ... \wedge L_n]$ *cubre en grado* σ *a una tupla* t cuando ésta satisface en grado σ a sus antecedentes:

$$C \text{ cubre en grado } \sigma \text{ a t} \Leftrightarrow \models_t(L_1 \wedge L_2 \wedge ... \wedge L_n, \sigma) \quad \text{[EC. 4.23]}$$

El *conjunto cubierto por una cláusula* C será un conjunto borroso, que denominaremos $\underline{T}_C(C)$, formado por todas las tuplas del conjunto de entrenamiento T cubiertas en algún grado por C. El grado de pertenencia de una tupla t al conjunto cubierto por C, es decir $\mu_{T_C(C)}(t)$, viene dado por el grado σ en que C cubre a t:

$$\underline{T}_C(C) = \{< t, \sigma >\} \quad \text{[EC. 4.24]}$$

siendo

$$\models_t(L_1 \wedge L_2 \wedge ... \wedge L_n, \sigma)$$

El *conjunto cubierto por una definición* D = [$C_1 \vee ... \vee C_m$] lo forma la unión de los conjuntos cubiertos por cada una de sus cláusulas:

$$\underline{T}_C(D) = \underline{T}_C(C_1) \cup ... \cup \underline{T}_C(C_m) \qquad \text{[EC. 4.25]}$$

4.3.2.5 Condición de k-cobertura y relación residual

Decimos que una cláusula C cumple la condición de *k-cobertura* o *cobertura borrosa* respecto de una tupla t cuando se cumple que:

$$\mu_{T_C(C)}(t) \geq k \cdot SB(t), \quad \text{para} \quad k \in [0, 1] \qquad \text{[EC. 4.26]}$$

es decir, cuando el grado con que C cubre a t es al menos k veces el grado con que t pertenece a la relación objetivo, siendo k el parámetro de k-cobertura.

La condición de k-cobertura se usa para determinar cuándo debe eliminarse una tupla t del conjunto de entrenamiento inicial T, tras finalizar la construcción de una cláusula C (apartado 4.3.3.5).

Al definir esta condición en función del signo borroso de la tupla, se obliga a que los mejores ejemplos de la relación objetivo deban ser cubiertos en mayor grado que los menos buenos (tuplas con signo borroso menor).

En el caso de conjuntos ordinarios, la condición de k-cobertura se reduce a la cobertura ordinaria para el valor k = 1.

En la versión actual de FZFOIL, el valor de k-cobertura se ha definido como un parámetro de entrada, modificable por el usuario, que por defecto vale 0.8.

La *relación residual* asociada a una cláusula C la representamos como $T_R(C)$ y la definimos como el subconjunto de tuplas de $\underline{P}$ (relación objetivo) para las que no se cumple la condición de k-cobertura:

$$\underline{T}_R(C) = \{<t, SB(t)> \mid \mu_{T_C(C)}(t) < k \cdot SB(t)\} \qquad \text{[EC. 4.27]}$$

Por tanto, la relación residual asociada a una cláusula será borrosa cuando la relación objetivo lo sea.

La *relación residual* de una definición D la forma la intersección de las relaciones residuales de sus cláusulas. Es también un conjunto borroso de tuplas (cuando $\underline{P}$ lo sea):

$$\underline{T}_R(D) = \underline{T}_R(C_1) \cap ... \cap \underline{T}_R(C_m) \qquad \text{[EC. 4.28]}$$

4.3.2.6 Condición de consistencia borrosa y k-consistencia

Decimos que una cláusula borrosa C = [L_0 :- $L_1 \wedge ... \wedge L_n$] es *consistente* sobre el conjunto T cuando se cumple la condición de *consistencia borrosa*, que definimos como:

$$\text{C es consistente sobre T} \quad \Leftrightarrow (\forall t \in T)\ (SB(t) \geq \mu_{T_C(C)}(t)) \qquad \text{[EC. 4.29]}$$

Esta condición es una generalización de la consistencia booleana ([EC. 2.12]), de modo que cumple el principio de extensión para conjuntos borrosos (apartado A.1.3.7).

También se puede definir una *k-consistencia* para cláusulas borrosas, función del conjunto cubierto $\underline{T}_C(C)$, de la relación objetivo $\underline{P}$ y del parámetro k, que toma valores en el intervalo [0, 1]:

$$\text{C es k-consistente sobre T} \Leftrightarrow \left|\underline{T}_C(C) \cap \underline{P}\right| \geq k \cdot \left|\underline{T}_C(C)\right| \qquad \text{[EC. 4.30]}$$

donde

$$\left|\underline{T}_C(C) \cap \underline{P}\right| = \sum_{t \in T} \min\left(\mu_{\underline{T}_C(C)}(t), SB(t)\right)$$

$$\left|\underline{T}_C(C)\right| = \sum_{t \in T} \left(\mu_{\underline{T}_C(C)}(t)\right)$$

Esta expresión es una extensión de la correspondiente a lógica booleana ([EC. 2.15]). Puede demostrarse (apéndice B, apartado B.1) que la condición de k-consistencia no es más que una generalización de la consistencia, que se cumple para todas las cláusulas consistentes y algunas inconsistentes (las de precisión[1] igual superior a k). Para k = 1 ambas son equivalentes.

Del mismo modo se puede definir la k-consistencia de una definición D:

$$\left|\underline{T}_C(D) \cap \underline{P}\right| \geq k \cdot \left|\underline{T}_C(D)\right|, \quad \text{para} \quad k \in [0, 1] \qquad \text{[EC. 4.31]}$$

En la versión actual de FZFOIL, se aplica la condición de k-consistencia, siendo k un parámetro configurable por el usuario, y por defecto vale 0.9.

Pueden encontrarse otras definiciones para la consistencia en el caso de conjuntos borrosos, por ejemplo en [González y Pérez, 95][2]:

$$\left|\underline{T}_C(C) \cap \overline{P}\right| \leq k \cdot \left|\underline{T}_C(C) \cap \underline{P}\right|, \quad \text{para} \quad k \in [0, 1] \qquad \text{[EC. 4.32]}$$

4.3.2.7 Condición de completitud borrosa y k-completitud

Decimos que una cláusula borrosa $C = [L_0 :- L_1 \wedge ... \wedge L_n]$ es *completa* sobre el conjunto T cuando se cumple la condición de *completitud borrosa*, que definimos como:

$$\text{C es completa sobre T} \quad \Leftrightarrow (\forall t \in T)\left(\mu_{\underline{T}_C(C)}(t) \geq SB(t)\right) \qquad \text{[EC. 4.33]}$$

Esta condición es una extensión borrosa de la completitud booleana ([EC. 2.17]).

---

1. Suponiendo que la precisión de una cláusula borrosa se define como:
$$\text{precisión} = \left|\underline{T}_C(C) \cap \underline{P}\right| \,/\, \left|\underline{T}_C(C)\right|$$
por extensión de la precisión definida en lógica booleana (ver nota en pág. 73).

2. Realmente definen una k-consistencia superior e inferior para las reglas borrosas, aplicando medidas de posibilidad y necesidad en los antecedentes y consecuente borrosos.

También se puede definir una condición de *k-completitud borrosa*, en función del conjunto cubierto $\underline{T}_C(C)$ y de la relación objetivo $\underline{P}$, y de un parámetro $k \in [0, 1]$.

$$C \text{ es k-completa sobre T} \Leftrightarrow \left|\underline{T}_C(C) \cap \underline{P}\right| \geq k \cdot \left|T \cap \underline{P}\right| \qquad \text{[EC. 4.34]}$$

donde

$$\left|\underline{T}_C(C) \cap \underline{P}\right| = \sum_{t \in T} \min\left(\mu_{T_C(C)}(t), SB(t)\right)$$

$$\left|T \cap \underline{P}\right| = \sum_{t \in T} \left(SB(t)\right)$$

Puede demostrarse que la condición de k-completitud borrosa es una generalización de la completitud borrosa, y que ambas son equivalentes para k=1.

También es sencillo demostrar que si una cláusula cumple la condición de k-cobertura respecto de todas las tuplas del conjunto T, entonces se cumple que C es k-completa sobre T (para $k \in [0, 1]$):

$$(\forall t \in T)\left(\mu_{T_C(C)}(t) \geq k \cdot SB(t)\right) \Rightarrow \left|\underline{T}_C(C) \cap \underline{P}\right| \geq k \cdot \left|T \cap \underline{P}\right| \qquad \text{[EC. 4.35]}$$

Por tanto, es inmediato deducir que una condición suficiente (pero no necesaria) para que una cláusula sea k-completa es que la relación residual asociada (apartado 4.3.2.5) sea vacía:

$$\underline{T}_R(C) = \varnothing \qquad \Rightarrow C \text{ es k-completa sobre T} \qquad \text{[EC. 4.36]}$$

El recíproco no es cierto en general (a diferencia de lo que ocurre en el caso booleano, [EC. 2.19]), pues sólo se puede asegurar que la relación residual es vacía cuando se cumple la condición de completitud (o k-completitud para k=1):

$$\underline{T}_R(C) = \varnothing \qquad \Leftarrow C \text{ es completa sobre T} \qquad \text{[EC. 4.37]}$$

Para una definición lógica borrosa D también se pueden definir las condiciones de completitud y k-completitud, para $k \in [0, 1]$:

$$D \text{ es completa sobre T} \Leftrightarrow (\forall t \in T)\left(\mu_{T_C(D)}(t) \geq SB(t)\right) \qquad \text{[EC. 4.38]}$$

$$D \text{ es k-completa sobre T} \Leftrightarrow \left|\underline{T}_C(D) \cap \underline{P}\right| \geq k \cdot \left|T \cap \underline{P}\right| \qquad \text{[EC. 4.39]}$$

En la actual versión de FZFOIL se aplica la condición de k-completitud, siendo k un parámetro configurable por el usuario que, por defecto, toma el valor 0.9.

En [González y Pérez, 95] puede verse otra definición de completitud borrosa, para finalizar la construcción de una definición. Se basa en la cardinalidad del corte-α (apartado A.1.4.3), tomando α como un umbral de cobertura mínima, aplicado sobre el conjunto de tuplas de P cubiertas por la última cláusula:

$$\left|{}^{\alpha}(P \cap T_C(C))\right| = 0 \qquad \text{[EC. 4.40]}$$

Por tanto, esta ecuación no representa realmente una medida de la completitud, sino más bien un criterio para finalizar la construcción de la definición.

4.3.2.8 Longitud de descripción extensional

Al extender el concepto de pertenencia en una relación ordinaria por el de grado de pertenencia en una borrosa, se hace necesario *borrosificar* también otros conceptos, como el de longitud de descripción extensional para una relación borrosa.

1. LDE de una relación

Como vimos en el apartado 2.5.4.5, la longitud de descripción extensional de una relación ordinaria Q se define como el número de bits necesarios para enumerar sus p tuplas en un conjunto de cardinalidad N ([EC. 2.21]); para ello, se informaba de:

- el tamaño N del conjunto universal para la relación Q:

  $\log_2(N)$ bits

- el número de posibles subconjuntos de cardinalidad p que existen en el conjunto universal (es decir, combinación de N elementos tomados de p en p):

  $\log_2(\binom{N}{p})$ bits

Para el caso borroso se puede utilizar una definición similar que, además, incluya información sobre los grados de pertenencia de las tuplas a la relación $\underline{Q}$:

- Si denominamos $\{n_0, n_1,..., n_L\}$ al conjunto de niveles (es decir, diferentes grados de pertenencia, según apartado A.1.4.4) de la relación $\underline{Q}$:

  $$\Lambda(\underline{Q}) = \{n_0, n_1,..., n_L\}, \quad \text{siendo} \quad 0=n_0 < n_1 <... < n_L \leq 1$$

  podemos definir los conjuntos de tuplas ${}^{[ni]}Q$, formados por las tuplas del conjunto universal que pertenecen a U con grado n_i:

  $${}^{[ni]}Q = \{t \in U \mid \mu_Q(t) = n_i\} = {}^{n_i}Q - {}^{n_{i+1}}Q, \ \forall i \in \{0,..., L\}$$

  siendo ${}^{ni}Q$ el corte alpha (apartado A.1.4.3) de Q para nivel n_i.

  Cada uno de los ${}^{[ni]}Q$ se puede codificar (siguiendo un razonamiento similar al del caso de Q ordinaria) con:

  $\log_2(N) + \log_2(\binom{N}{\left|{}^{[n_i]}Q\right|})$ bits

- Si tenemos en cuenta que cada tupla sólo puede pertenecer a un subconjunto ${}^{[ni]}Q$, podemos descontar de N las tuplas consideradas en los subconjuntos anteriores; así, para codificar un ${}^{[ni]}Q$ cualquiera harán falta:

  $\log_2(N_i) + \log_2(\binom{N_i}{\left|{}^{[n_i]}Q\right|})$ bits

  siendo

  $$N_1 = N, \qquad N_i = N - \sum_{j=1}^{i-1} \left|{}^{[n_j]}Q\right|, \ i=\{2...L\}, \qquad 0=n_0 < n_1 <... < n_L \leq 1$$

La expresión resultante para la longitud de descripción extensional de una relación borrosa $\underline{Q}$, que representamos como LDE($\underline{Q}$), se calculará como la suma de las longitudes de descripción para los subconjuntos ${}^{[ni]}Q$. De todos los subconjuntos ${}^{[ni]}Q$ sólo codificaremos aquellos que

correspondan a tuplas con grado de pertenencia a $\underline{Q}$ distinto de cero, es decir, no hará falta codificar ${}^{[0]}Q$:

$$LDE(\underline{Q}) = \sum_{i=1}^{L} (\log_2(N_i) + \log_2(\binom{N_i}{\left|{}^{[n_i]}Q\right|})) \text{ bits} \qquad \text{[EC. 4.41]}$$

siendo

$$N_1 = N, \qquad N_i = N - \sum_{j=1}^{i-1} \left|{}^{[n_j]}Q\right|, \; i=\{2...L\}$$

N el número de tuplas del conjunto universal para $\underline{Q}$

$0=n_0 < n_1 <... < n_L \leq 1$ el conjunto de niveles de $\underline{Q}$

${}^{[ni]}Q$ el conjunto ordinario de tuplas del nivel i-ésimo (n_i) de $\underline{Q}$

Podemos comprobar que esta expresión es una extensión borrosa de la [EC. 2.21], pues tomando ${}^{[1]}Q$ como el subconjunto de tuplas "+" de una relación ordinaria, y ${}^{[0]}Q$ como el de tuplas "–", coinciden ambas expresiones.

2. LDE de cláusulas y definiciones

Se puede extender la definición de LDE para aplicarla a una cláusula C (o una definición D) borrosa (cuyo conjunto cubierto sea borroso):

$$LDE(C) = \sum_{i=1}^{L} (\log_2(N_i) + \log_2(\binom{N_i}{\left|{}^{[n_i]}Q\right|})) \text{ bits} \qquad \text{[EC. 4.42]}$$

$$LDE(D) = \sum_{i=1}^{L} (\log_2(N_i) + \log_2(\binom{N_i}{\left|{}^{[n_i]}Q\right|})) \text{ bits} \qquad \text{[EC. 4.43]}$$

donde L se refiere al número de niveles del conjunto cubierto $\underline{T}_C(C)$ (o $\underline{T}_C(D)$) y los ${}^{[ni]}Q$ se construyen a partir del mismo.

4.3.2.9 Longitud de descripción intensional

La longitud de descripción intensional de un literal borroso se puede definir de modo parecido a como se hizo para un literal ordinario ([EC. 2.24]):

$$LDI(L_i) = \log_2(NE) + \log_2(NR) + \log_2(NA) \text{ bits} \qquad \text{[EC. 4.44]}$$

donde el primer sumando indica la etiqueta lingüística (apartado 4.3.4), el segundo el predicado (suponiendo que hay NR relaciones en la base de datos) y el último los argumentos del literal (suponiendo NA posibles argumentos). En el caso de literales ordinarios, las dos únicas etiquetas lingüísticas posibles (NE) son el signo afirmativo o negativo del literal.

Para cláusulas y definiciones borrosas, son aplicables las mismas expresiones para LDI indicadas en el caso booleano ([EC. 2.25] y [EC. 2.26]).

4.3.3 Algoritmo de FZFOIL

Al igual que en FOIL, en FZFOIL se construyen las definiciones siguiendo un algoritmo formado por dos bucles:

- un bucle externo, en el que se construyen definiciones lógicas completas y consistentes,
- un bucle interno, en el que se construyen cláusulas consistentes.

También se utiliza un heurístico (tanto al construir definiciones como cláusulas) para limitar la complejidad de las descripciones, permitiendo la construcción de definiciones inconsistentes y/o incompletas. Este heurístico se basa en el principio de *Mínima Longitud de Descripción* de Rissanen ([EC. 2.31]).

Sin embargo, las diferencias en el algoritmo son numerosas ya que, al trabajar con lógica borrosa, ha sido necesario extender muchos de los conceptos utilizados en FOIL:

- El signo "+" o "–" de las tuplas se sustituye por el signo borroso (apartado 4.3.2.2), que representaremos como SB(t) para una tupla t.
- En general, las tuplas de cada conjunto de entrenamiento sólo satisfarán en cierto grado los literales borrosos candidatos.
- Los conjuntos de entrenamiento intermedios serán también borrosos, por lo que sus tuplas tendrán un grado de pertenencia asociado.
- Se necesitan nuevos criterios para decidir cuándo es consistente una cláusula, en función de los grados de pertenencia y de los signos borrosos de las tuplas del conjunto de entrenamiento (apartado 4.3.2.6).
- También se necesita una nueva definición para definiciones completas (apartado 4.3.2.7).
- Deberá establecerse un criterio a la hora de eliminar del conjunto de entrenamiento inicial (T) las tuplas cubiertas por una cláusula, para formar el conjunto de entrenamiento para la siguiente cláusula (condición de k-cobertura, apartado 4.3.2.5).
- Deberá definirse la longitud de codificación de una relación borrosa (apartado 1) y de un literal borroso (apartado 4.3.2.9), para poder aplicar el principio de mínima longitud de descripción.

4.3.3.1 Bucle externo: construcción de definiciones k-completas

En el bucle externo de FZFOIL se construyen definiciones completas, formadas por una cláusula de Horn o por la disyunción de varias de ellas. En cada iteración del bucle se añade una nueva cláusula a la definición en construcción y se modifica el conjunto de entrenamiento, eliminando del mismo las tuplas cubiertas por la nueva cláusula. Esta idea es, esencialmente, la misma que se utiliza en otros algoritmos bien conocidos en el campo del aprendizaje, como el sistema AQ ([Michalski, 87]).

El algoritmo finaliza cuando se cumple la condición de k-completitud, concluyendo con una definición k-completa, o cuando se cumple la condición de poda por mínima longitud de descripción, y entonces la definición es incompleta.

Definición& FZFOIL (T^+, T^-, $T^\sim$, k-comp, k_cons, p, q_1, q_2,...)

{

Definicion D^0 = FALSE;

```
    Conjunto T_C^0 = Ø;
    Conjunto T_R^0 = T ∩ P;
    i = 1;
    repetir
        Clausula C_i = construyeC (T_R^(i-1), T⁻, k_cons, p, q_1, q_2,...);
        D^i = D^(i-1) ∨ C_i;            /* añade cláusula k-consistente */
        T_C^i = T_C^(i-1) ∪ T_C(C_i);   /* actualiza T_C */
        T_R^i = T_R^(i-1) ∩ T_R(C_i);   /* actualiza T_R */
        i ++;
    hasta que (D^(i-1).es_Kcompleta(k_comp, T_C^(i-1), P) || poda(D^(i-1), T_C^(i-1)));
    return D^(i-1);
}.
```

El algoritmo finaliza cuando se cumple la condición de completitud borrosa, con una definición k-completa, o cuando se produce la poda por criterio de mínima longitud de descripción, resultando entonces una definición incompleta.

Las diferencias con el algoritmo basado en relaciones ordinarias (apartado 2.5.3.1) son:

- Tanto el conjunto cubierto por la cláusula ($T_C(C_i)$), como el cubierto por la definición (T_C^i) y la relación residual asociada (T_R^i) son conjuntos borrosos.
- La relación residual asociada a la definición ya no se puede calcular como la diferencia entre el conjunto inicial T_R^0 y el conjunto T_C^i cubierto por la definición, como se hacía en el caso booleano; es decir, la [EC. 2.10] deja de ser una igualdad:

  $$T_R^i \approx T_R^0 - T_C^i \qquad \text{[EC. 4.45]}$$

  ahora hay que aplicar la condición de k-cobertura para eliminar del conjunto inicial de entrenamiento las tuplas cubiertas por una cláusula.
- Por último, la condición de completitud se sustituye por la de k-completitud borrosa, como se indica en el apartado 4.3.2.7.

4.3.3.2 Bucle interno: construcción de cláusulas k-consistentes

En el bucle interno de FZFOIL (función *construyeC* en el pseudocódigo del bucle externo) se construyen cláusulas k-consistentes, cuyo consecuente es el predicado objetivo aplicado sobre variables, y cuyos antecedentes son literales construidos con predicados de la base de datos, o con predicados predefinidos (como la igualdad, comparación, etc. entre atributos).

En cada iteración del bucle interno se añade un nuevo antecedente a la cláusula y se modifica el conjunto de entrenamiento, seleccionando las tuplas que satisfacen el nuevo literal. La construcción de la cláusula finaliza cuando ésta es consistente o se cumple la condición de poda por mínima longitud de descripción (entonces se obtiene una cláusula inconsistente).

```
Clausula& construyeC (T_R, T⁻, k_cons, p, q_1, q_2,...)
{
    Clausula C^0 = p;                    /* cláusula inicial = "p :- TRUE" */
    Conjunto T_1 = construyeT(T_R) ∪; T⁻;
```

```
    i = 1;
    repetir
        Literal L_i = buscaAntecedente (T_i, p, q_1, q_2,...);
        C^i = C^(i-1) ∨ (¬L_i);                    /* añade antecedente L_i */
        T_(i+1) = actualizarT (T_i, L_i);          /* tuplas que satisfacen L_i */
        T_C = proyectarT_C (T_1, T_(i+1));         /* tuplas que satisfacen C^i */
        i ++;
    hasta que (C^(i - 1).es_Kconsistente(k_cons, T_C, P) || poda(C^(i - 1), T_1));
    return C^(i - 1);
};
```

Las diferencias respecto a la versión con relaciones ordinarias (apartado 2.5.3.2) son:

- El conjunto inicial de entrenamiento para una cláusula (T_1) se construye con las tuplas de la relación residual T_R, a las que se asigna inicialmente un grado de pertenencia unidad (siguiendo el razonamiento expresado en [EC. 4.49]). En este conjunto inicial también se incluyen las tuplas negativas del conjunto T^-, también con grado de pertenencia unidad:

$$T_1 = \{<t, 1> \mid \mu_{T_R}(t) > 0\} \cup T^- \qquad \text{[EC. 4.46]}$$

- La función *actualizarT* crea un nuevo conjunto local de entrenamiento T_{i+1}, seleccionando las tuplas de T_i que satisfacen en algún grado al literal L_i (conservando cada nueva tupla de T_{i+1} el signo borroso de su antecesor). Cuando L_i introduce nuevas variables, las tuplas de T_{i+1} expanden su tamaño respecto de las de T_i; en ese caso, una tupla de T_i puede dar lugar a más de una en T_{i+1}. El grado de pertenencia de una tupla t al nuevo conjunto T_{i+1} viene dado por el mínimo entre su grado de pertenencia a T_i y su grado de satisfacción al literal L_i ([EC. 4.54]).
- La condición de consistencia se ha sustituido por la de k-consistencia borrosa, definida en el apartado 4.3.2.6.

4.3.3.3 Evaluación de literales

El objetivo de la función *buscaAntecedente* (ver pseudocódigo de apartado 4.3.3.2) es explorar el espacio de los literales candidatos y evaluarlos para elegir el mejor entre ellos.

Para medir la bondad de cada literal, Quinlan utiliza en FOIL un heurístico (*Ganancia*) basado en una medida de la información (ecuación [EC. 2.29]). Sin embargo, este heurístico presenta algunas deficiencias en la evaluación de literales (apartado 3.2), por lo que en FZFOIL se ha preferido partir de otros heurísticos, basados en el interés de una regla (apartado 3.3), con los que se superan estos problemas.

Además, resulta más interesante que los argumentos de la función de evaluación no se midan en los conjuntos de entrenamiento locales (creados durante la construcción de una cláusula), sino en el conjunto inicial, ya que sólo de este modo se pueden prevenir algunos de los errores detectados (apartado 3.3.3). Por ello, se realizarán proyecciones de los conjuntos de entrenamiento locales sobre el conjunto inicial, de forma similar a como vimos para las relaciones ordinarias (figura 3-6).

Cada vez que se añade un nuevo literal L_i a una cláusula en construcción C^{i-1} se genera un nuevo conjunto de entrenamiento T_{i+1}, con las tuplas de T_i que satisfacen a L_i. La proyección del nuevo

conjunto T_{i+1} sobre T_1 la representamos como $T_1^{[i+1]}$, y será siempre un subconjunto de $T_1^{[i]}$ (proyección de T_i sobre T_1):

$$T_1^{[i+1]} = \{< t, \sigma > \mid (t \in T_1) \text{ y } \models_t (L_1 \wedge ... \wedge L_{i+1}, \sigma)\} \subseteq T_1^{[i]} \qquad \text{[EC. 4.47]}$$

Si alguno de los literales L_i es borroso, el conjunto de entrenamiento local T_{i+1} que se genera será también borroso. En este caso, la proyección de este conjunto (y de los posteriores) sobre T_1, es decir $T_1^{[i+1]}$, será también un subconjunto (borroso) de T_1.

El interés de una cláusula debe ser máximo cuando el conjunto (borroso) cubierto por ella sea igual que la relación objetivo o, dicho de otro modo, cuando para cada tupla del conjunto cubierto coincidan su grado de pertenencia y su signo borroso. Por ello, es conveniente que la función de evaluación mida la correlación entre estos dos parámetros de cada tupla. Esto se puede conseguir modificando los argumentos de la función RI ([EC. 3.6]), para conseguir una nueva función de evaluación que podemos llamar FRI (*Fuzzy Rule Interest*):

$$FRI(N, N_A, N_B, N_{A \wedge B}) = \frac{N_{A \wedge B} - (N_A \cdot N_B / N)}{\sqrt{N_A \cdot N_B \cdot (1 - N_A/N) \cdot (1 - N_B/N)}} \qquad \text{[EC. 4.48]}$$

siendo

$N = \lvert T_1 \rvert$	cardinalidad del conjunto de entrenamiento T_1
$N_A = \lvert T_1^{[i+1]} \rvert = \sum_t \mu_{T_1^{[i+1]}}(t)$	cardinalidad escalar (apartado A.1.4.5) del conjunto cubierto por la cláusula ($T_1^{[i+1]}$)
$N_B = \sum_{t \in T_1} SB(t)$	suma de los signos borrosos de las tuplas del conjunto de entrenamiento T_1
$N_{A \wedge B} = \sum_t \min(SB(t), \mu_{T_1^{[i+1]}}(t))$	suma del mínimo entre signo borroso y grado de pertenencia al conjunto cubierto por la cláusula.

4.3.3.4 Conjuntos de entrenamiento para relaciones borrosas

El conjunto de entrenamiento T que construye FOIL, para inducir una definición de una relación R ordinaria, está formado por dos subconjuntos: T^+, con las tuplas que pertenecen a R (tuplas positivas o ejemplos), y T^-, con las tuplas que no pertenecen a R (tuplas negativas o contraejemplos de R).

En el caso de FZFOIL, supondremos que el conjunto de entrenamiento inicial (T) es también un conjunto ordinario, pues realmente se define como el conjunto cubierto (apartado 4.3.2.4) por la cláusula de Horn C^0, sin literales negativos:

$$C^0 = [L] \Leftrightarrow \quad C^0 = [L \text{ :- TRUE}] \Rightarrow \quad \mu_T(t) = 1, \forall t \qquad \text{[EC. 4.49]}$$

siendo L el literal construido para la relación objetivo R. Por tanto, T será un conjunto ordinario en el que todas sus tuplas tienen grado de pertenencia igual a 1. El signo de las tuplas dependerá de la relación objetivo R: si R es ordinaria, los signos serán "+" y "–"; en el caso borroso, tendrán signos borrosos.

En cualquier caso, podemos seguir considerando que los conjuntos T^+ y T^- son subconjuntos de T, formados por las tuplas con signo borroso igual a 1 y 0 respectivamente:

$$T^+ = \{t \mid SB(t) = 1\} \qquad \text{[EC. 4.50]}$$

$$T^{-} = \{t \mid SB(t) = 0\} \quad \text{[EC. 4.51]}$$

Además, en el caso de una relación objetivo $\underline{R}$ borrosa, habrá que considerar también el subconjunto de tuplas con signo borroso distinto de 0 y de 1:

$$T^{\sim} = \{t \mid 0 < SB(t) < 1\} \quad \text{[EC. 4.52]}$$

De este modo, el conjunto de entrenamiento se construiría como la unión de estos tres conjuntos:

$$T = T^{+} \cup T^{-} \cup T^{\sim} \quad \text{[EC. 4.53]}$$

Por tanto, en el caso de relaciones borrosas, este conjunto de entrenamiento inicial (T) es un conjunto ordinario, al igual que en el caso de relaciones ordinarias.

Las tuplas negativas (conjunto T^{-}) del conjunto de entrenamiento pueden venir explícitamente definidas, o ser construidas con la asunción de mundo cerrado (suponiendo que el grado de pertenencia de las tuplas que no aparecen en la relación $\underline{R}$ es cero).

Al igual que se hace en FOIL, además del conjunto de entrenamiento inicial T, se construyen una serie de conjuntos de entrenamiento locales, que denominaremos T_i, para evaluar los literales candidatos durante la construcción de las cláusulas (apartado 2.5.3.2). En el caso de relaciones borrosas, estos conjuntos locales serán borrosos, ya que aunque se construyen a partir del conjunto inicial T, sus tuplas se seleccionan de acuerdo con su grado de satisfacción (apartado 4.3.2.3) a los literales antecedentes de la cláusula, como veremos más adelante.

4.3.3.5 Modificación de los conjuntos de entrenamiento

- El conjunto de entrenamiento inicial para una definición lo forman las tuplas de la relación objetivo más las tuplas que no pertenecen a la misma (definidas explícitamente por el usuario u obtenidas por la suposición de mundo cerrado).

  La actualización de la relación residual asociada a la definición se realiza eliminando las tuplas del conjunto cubierto que cumplen la condición de k-cobertura.

- Para las cláusulas, se crea un conjunto inicial de entrenamiento T_1 con las tuplas de la relación residual asociada a la definición (tuplas que no cumplen la condición de k-cobertura definida en el apartado 4.3.2.5). Las tuplas de este conjunto T_1 pertenecen al mismo con grado de pertenencia unidad; su signo borroso es igual a su grado de pertenencia a la relación objetivo.

  Los conjuntos de entrenamiento posteriores T_i (para evaluar el resto de antecedentes de una cláusula) se construyen a partir del primero, modificando el grado de pertenencia de sus tuplas, y manteniendo el signo borroso de las mismas. El grado de pertenencia para una tupla se calcula como el mínimo entre el grado de pertenencia al conjunto anterior y el nivel en que satisface al literal candidato:

  $$\mu_{T_{i+1}}(t) = \min\{\mu_{T_i}(t), \sigma\}, \text{ siendo } \models_t(L_i, \sigma) \quad \text{[EC. 4.54]}$$

  Para una tupla t, el grado de pertenencia al conjunto proyectado $T_1^{[i+1]}$, en el caso general de que t se expanda en varias tuplas {t', t'',...} en T_{i+1}, se calculará como el máximo de los grados de pertenencia a T_{i+1} de sus tuplas descendientes:

  $$\mu_{T_1^{[i+1]}}(t) = \max\{\mu_{T_{i+1}}(t'), \mu_{T_{i+1}}(t''), \ldots\} \quad \text{[EC. 4.55]}$$

4.3.4 Invención de literales borrosos: etiquetas lingüísticas

En FZFOIL se permite la construcción de definiciones con literales borrosos, tal como hemos visto en el apartado 4.3.3.3, aplicando directamente sobre ellos la función de evaluación representada en [EC. 4.48], que no es más que una generalización de la que se usaba en FOIL para literales ordinarios ([EC. 3.8]).

Pero pueden obtenerse ventajas adicionales del uso de literales borrosos mediante la utilización de *etiquetas lingüísticas* (apartado A.5). Estas etiquetas actúan sobre un literal borroso como modificadores de su significado, entendiendo éste como el grado en que cada tupla lo satisface. Las etiquetas modifican el grado de satisfacción de las tuplas del conjunto de entrenamiento para el literal en cuestión, es decir, tienen un comportamiento similar al descrito en apartado 4.1.2.7.

Las etiquetas aplicables sobre un literal borroso pueden ser:

- modificadores fuertes (como “muy”, “sumamente”, “bastante”, etc.), que refuerzan el significado asociado al literal, de modo que las tuplas lo satisfacen en menor grado;
- modificadores débiles (como “algo”, “más o menos”, etc.), que relajan el significado del literal, de modo que las tuplas lo satisfarán en mayor grado;
- modificador identidad, es decir, literal sin ninguna etiqueta lingüística que modifique su significado.

Como ya hemos dicho, el resultado de aplicar una etiqueta lingüística sobre un literal borroso L, se refleja como una modificación del grado en que las tuplas del conjunto de entrenamiento lo satisfacen. Si denominamos h a la función de modificación asociada a una etiqueta H (generalmente h es de la forma $h_\alpha(a) = a^\alpha$, según [EC. A.48]), el grado en que cualquier tupla t satisface al literal con etiqueta (que representaremos como HL) será:

$$\models_t(HL, h(\sigma)), \text{ siendo } \models_t(L, \sigma) \qquad \text{[EC. 4.56]}$$

Si nos fijamos en los parámetros de la función de evaluación usada en FZFOIL ([EC. 4.48]), podemos comprobar que los parámetros que se ven afectados por el uso de etiquetas lingüísticas en un literal borroso son N_A y $N_{A \wedge B}$, es decir, los que dependen del significado del literal borroso (a través del grado de pertenencia a $T_1^{[i+1]}$ según [EC. 4.54] y [EC. 4.55]). El nuevo valor de estos parámetros se calculará del mismo modo, pero modificando el grado de pertenencia al nuevo conjunto (resultado de aplicar [EC. 4.56] sobre [EC. 4.55] y [EC. 4.54]):

$$\mu_{T_1^{[i+1]}}(t) = \max\{ \min(\mu_{T_i}(t^j), h(\sigma)), \forall t^j \text{ extensión de t en } T^i \}, \text{ siendo } \models_t(L_i, \sigma) \qquad \text{[EC. 4.57]}$$

También tiene interés la aplicación de la negación lógica sobre literales borrosos con etiquetas lingüísticas. Así para un literal L podrían construirse, por ejemplo, los literales “no muy L”, “no bastante L”, “no algo L”, etc. En este caso, el grado de satisfacción de una tupla t para la negación de un literal L con la etiqueta H (representado como ¬HL) se calculará como:

$$\models_t(\neg HL, 1 - h(\sigma)), \text{ siendo } \models_t(L, \sigma) \qquad \text{[EC. 4.58]}$$

Por tanto, el resultado de permitir literales borrosos con etiquetas lingüísticas se traduce en una extensión del espacio de búsqueda de literales, a cambio de la posibilidad de construir mejores definiciones borrosas.

En la versión actual de FZFOIL se han implementado tres etiquetas lingüísticas, definidas del siguiente modo:

H_2 = “muy” , siendo $h_2(a) = a^2$

$H_{1.5}$ = “bastante” , siendo $h_{1.5}(a) = a^{1.5}$

$H_{0.5}$ = “algo” , siendo $h_{0.5}(a) = a^{0.5}$

de modo que por cada literal borroso L sin etiquetas que se evalúe, se evaluarán 6 literales adicionales, resultantes de aplicar cada una de estas etiquetas sobre L (con los literales negados correspondientes). A continuación veremos un ejemplo en el que se aplican estas etiquetas.

4.3.5 Relaciones intensionales

Las relaciones intensionales son relaciones definidas por el usuario, pero que no son introducidas a través de su conjunto de tuplas, sino mediante una expresión lógica formada por una cláusula de Horn. Su aspecto es, por tanto, similar al de las definiciones lógicas inducidas por FZFOIL.

De este modo, es posible introducir fácilmente conocimiento de base en el sistema (ver nota a pie de pág. 56), construido a partir de las relaciones extensionales o de otras relaciones intensionales definidas previamente.

El consecuente de la cláusula que define una relación intensional es el predicado asociado a dicha relación, con todos sus atributos variables. Los antecedentes son literales correspondientes a:

- relaciones extensionales de la base de datos;
- o bien relaciones intensionales definidas previamente;
- o bien relaciones predefinidas como la igualdad entre variables, o la comparación, etc.

Por ejemplo, dadas las relaciones extensionales de los *matrimonios* y de la relación *hermanos*, podríamos construir de forma intensional la relación binaria de los *cuñados*:

cuñados(A, B) :- matrimonio(A, C), hermanos(C, B)

4.4 Ejemplo

Veamos con un pequeño ejemplo el funcionamiento de la versión actual de FZFOIL. Supongamos una base de datos relacional en la que tenemos las siguientes relaciones:

Tabla 4-1: enfermo

b3 : 0.9	b4 : 0.72	c3 :0.7	c4 :0.8	c5 :0.88
d2 :0.9	d3 :0.85	d4 : 0.75		

Tabla 4-2: padre_de

a1, b1	a1, b2	a2, b1	a2, b2	a3, b3

Tabla 4-2: padre_de

a3, b4	a4, b3	a4, b4	b1, c1	b1, c2
b2, c3	b3, c3	b4, c4	b4, c5	c1, d1
c2, d1	c2, d2	c2, d3	c4, d2	c4, d3
c5, d4				

Tabla 4-3: jefe_de

a1,b1	a2,b2	a2,b3	a3,b2	a3,b3
a4,b3	b1,c1	b2,c2	b2,c3	b2,c4
b3,c2	b3,c3	b3,c4	b4,c4	b4,c5
c1,d1	c2,d1	c3,d2	c3,d3	c3,d4
c4,d2	c4,d3	c4,d4	c5,d4	

Tabla 4-4: fumador

b3 :0.95	b4 :0.8	d4 :0.75	d2 :0.4	c1 :0.2
a2 :0.3				

Tabla 4-5: barbudo

a3 :0.4	a4 :0.7	b4 :0.88	c1 :0.9	c2 :0.5
c4 :0.95	d1 :0.6	d2 :1		

Las tuplas en las que aparece el carácter ":" seguido de un número real entre 0 y 1 son tuplas de una relación borrosa ("enfermo", "fumador" y "barbudo" en este ejemplo); el número indica el grado de pertenencia a dicha relación.

Queremos inducir una definición para la relación borrosa "enfermo", usando para ello cualquiera de las relaciones (borrosas o no) de nuestra base de datos. En la versión original de FOIL, este ejemplo sólo podría ejecutarse para relaciones ordinarias, ya que no podríamos representar los diferentes grados de pertenencia de las tuplas. En nuestra versión, además, permitimos más matices (más flexibilidad) en la representación de las definiciones inducidas, al poder inventar nuevos predicados a partir de los existentes.

La definición que se induce (D1) está formada por la disyunción de dos cláusulas (C1 y C2):

```
[D1] {8.951 bits} TC: [6.58/7.04]; TR: [0.12]
    [C1] {7.953 bits} [6.70/19.00]-> [5.04/5.37]
        enfermo (A) :-
            padre_de (B A),
            ALGO_enfermo (B).
    [C2] {3.584 bits} [1.62/13.00]-> [1.54/1.67]
```

enfermo (A) :-

MUY_fumador (A).

Los números que aparecen en la primera línea indican:

- La longitud de descripción de la definición: 8.366 bits. Este número indica la longitud en bits necesarios para representar la definición. Este valor no debe superar la longitud de descripción explícita (conjunto de tuplas) de la relación objetivo, pues se busca siempre una definición más compacta que la enumeración de todas las tuplas.
- El valor TC (6.50/7.04) es una medida de la consistencia de la definición, pues indica la relación entre las tuplas "positivas" cubiertas respecto del total de tuplas cubiertas (ver apartado 4.3.2.6).
- El valor TR (0.37) es el número de tuplas "positivas" no cubiertas por la definición (para definiciones totalmente completas, este valor es 0).

Para cada cláusula se indica:

- Longitud de descripción, de modo similar a como se hace para la definición global.
- Una medida de la bondad de la cláusula, pues se indica el número de tuplas "positivas" respecto del total del conjunto de entrenamiento, y a continuación el número de tuplas "positivas" cubiertas respecto del total de ellas cubiertas. Por ejemplo, la cláusula C1 ha sido inducida con un conjunto de entrenamiento formado por 19 tuplas, de las que 6.87 eran "positivas", y ha cubierto un total de 5.34 tuplas, de ellas 4.90 positivas.

Se puede observar que en la segunda cláusula se ha inventado un nuevo predicado, al añadir la etiqueta "bastante" al predicado borroso "fumador". De este modo, podemos leer la definición como: "una persona está enferma cuando es hija de otra que está enferma, o bien, cuando es bastante fumadora".

4.5 Conclusiones

La introducción de relaciones intensionales, además de las relaciones extensionales, ofrece más potencia en la representación del conocimiento de base.

Pero la aportación más interesante de FZFOIL es la introducción de relaciones borrosas en la base de datos. Las relaciones borrosas usadas en FZFOIL permiten una descripción más precisa del mundo real, al poder representar la incertidumbre asociada a muchos aspectos de la realidad, al menos desde el punto de vista humano. Sentencias como

"si tienes mucho frío, es que tienes fiebre"

"si tienes fiebre y te duelen los huesos, es probable que tengas gripe",

son difícilmente inducibles a partir de una base de datos con relaciones ordinarias, pues, se pierde exactitud al representar conceptos *borrosos* como la sensación de dolor o de frío mediante relaciones ordinarias, a las que las tuplas pueden pertenecen o no, pero las que pertenecen lo hacen todas en la misma medida. Sólo con relaciones borrosas se puede cuantificar el grado de pertenencia de las tuplas.

Por otro lado, la posibilidad de inventar predicados añadiendo adverbios de cantidad como mucho, bastante, etc. a las relaciones borrosas ya existentes ofrece la flexibilidad necesaria para poder inducir en FZFOIL definiciones como las anteriores.

La adaptación del algoritmo FOIL para poder manejar relaciones borrosas ha requerido una serie de cambios, orientados al manejo de los grados de pertenencia de las tuplas a sus relaciones borrosas. Quizá los más significativos hayan sido los realizados en la función de evaluación de los literales candidatos, y la redefinición de conceptos como completitud, consistencia o cobertura para datos borrosos. Para la invención de predicados borrosos, a partir de otros ya existentes en los datos de partida, se han definido curvas de pertenencia para cada una de las etiquetas permitidas en los mismos, en función de los grados de pertenencia de las tuplas a los conjuntos borrosos.

Capítulo 5:

ANÁLISIS DE RESULTADOS

5.1 Limitaciones

Todo sistema de aprendizaje lleva asociada una serie de limitaciones que restringen su ámbito de aplicación a problemas concretos, en los que los datos de entrada deben estar expresados de una forma característica y los resultados se ajustan a cierto formato.

Entre los lenguajes empleados por los sistemas de aprendizaje simbólico destaca la lógica de predicados de primer orden, pues su gran potencia descriptiva y su flexibilidad la hacen adecuada para describir, de una forma sencilla, estructuras y relaciones complejas existentes en gran número de problemas. Por ello, sistemas como FOIL y FZFOIL, que siguen el enfoque simbólico, inducen reglas que son adecuadas para describir soluciones complejas de problemas muy diversos. Sin embargo, hay que destacar dos limitaciones importantes en la utilización de la lógica de predicados en estos sistemas:

- Versión restringida de la lógica de predicados.

  Realmente se utiliza una versión limitada de la lógica de predicados, tanto en FOIL como en su versión borrosa FZFOIL. De hecho, no se permite la utilización de funciones ni en las definiciones lógicas que se construyen, ni en la definición de los predicados intensionales de entrada. Además, como es habitual en programación lógica, la semántica de la negación en las cláusulas de Horn es distinta de la semántica de la lógica de predicados.

- Vocabulario limitado

  No se permite construir nuevos predicados en las definiciones.

Por otro lado, existen limitaciones intrínsecas al propio algoritmo de inducción:

- Adquisición de conceptos pero no descubrimiento

  FOIL y FZFOIL son sistemas de adquisición de conceptos, pero no de descubrimiento o formación de conceptos. Es decir, es necesario especificar la relación objetivo para la que se quiere inducir una definición, pues estos sistemas no son capaces de *formar conceptos* existentes entre los datos, pero desconocidos de antemano.

- Problemas heredados de FOIL:

- Máximos locales en la función de búsqueda de literales, aunque este problema se corrige en gran medida en FZFOIL, mediante la utilización de nuevas funciones de evaluación de literales, basadas en medidas del interés.
- Sensibilidad al ruido en las definiciones recursivas, etc.

La extensión de la lógica de predicados hacia una lógica borrosa de predicados es un paso adelante en la flexibilidad y potencia descriptiva, tanto en la entrada de datos (mediante relaciones borrosas) como en la representación del conocimiento inducido (reglas borrosas). Esto aumenta el campo de aplicación de FZFOIL respecto de FOIL, especialmente al manejar conceptos borrosos, como los utilizados habitualmente por los humanos, ya que se facilita el modelado del problema y resulta más fácil de entender la solución del mismo (la definición del mismo). Sin embargo, existen otros tipos de incertidumbre, distintos de la borrosidad, que no se pueden representar en FZFOIL:

- Introducción de incertidumbre sólo mediante relaciones borrosas

  Existen diferentes métodos para representar la incertidumbre; en FZFOIL sólo se utiliza uno de ellos: la teoría de conjuntos borrosos, aplicada a la construcción de BD relacionales.

  Por otro lado, las reglas que se inducen representan conocimiento borroso (mediante literales borrosos) que, además, pueden tener asociado un factor de probabilidad.

5.2 Ámbito de aplicación

Entre las aplicaciones más inmediatas de FZFOIL se encuentra la construcción de definiciones lógicas para relaciones de una BD, al igual que ocurre en FOIL. En FZFOIL se amplía el campo de aplicación al permitir manejar relaciones borrosas, es decir, puede ser aplicado tanto sobre las tradicionales BD relacionales como sobre BD relacionales borrosas.

La utilidad de esta aplicación cada vez es más evidente, dado el interés que están adquiriendo los sistemas de control borrosos, sistemas expertos borrosos, etc.

Por otro lado, también resulta interesante simplemente para analizar datos y extraer conocimiento de los mismos. La utilización de la lógica borrosa de predicados en FZFOIL facilita la comprensión del conocimiento inducido, al manejar relaciones borrosas, más parecidas a los conceptos utilizados en lenguaje natural.

Aunque la introducción de la lógica borrosa en FZFOIL aumenta la complejidad de la búsqueda, resulta aplicable no sólo sobre pequeños conjuntos de entrenamiento, sino también sobre un conjunto voluminoso de datos reales, como se muestra a continuación.

5.3 Ejemplo de aplicación sobre datos reales: Proyecto SEIC

En este apartado se presentan algunos resultados obtenidos por FZFOIL en el Proyecto SEIC (*Servicio de Información Ciudadana*, PC-183), dentro de la línea de “Acciones Especiales PASO”, gestionada por el C.D.T.I. y cofinanciada por la Unión Europea y el Ministerio de Industria de España. Este proyecto ha estado coordinado por EnWare, S.A., y en él han

participado también, además del DIT, el Consorcio Regional de Transportes de Madrid, el Colegio Oficial de Farmacéuticos de Madrid y Genasys II Spain.

El proyecto SEIC ha servido como motor y campo de pruebas para nuestro sistema FZFOIL, al aportar no sólo un objetivo concreto (aunque no el principal) para el desarrollo del trabajo realizado, sino también una valiosa fuente de datos reales, con los que evaluar los resultados de manera más objetiva.

El objetivo del proyecto SEIC consiste en la construcción de un sistema de información de carácter general, orientado a los ciudadanos, que podrán acceder al mismo a través de unos puestos de información ciudadana (PICs) situados en la vía pública, normalmente en estaciones de Metro o en lugares de interés turístico. La información que se ofrecerá inicialmente será sobre transportes públicos y sobre farmacias, aunque, virtualmente, podrá extenderse a otros tipos de información.

Dentro del proyecto existen diferentes módulos, para cubrir los diferentes requisitos del mismo: base de datos sobre transportes, datos sobre farmacias, comunicaciones entre el módulo central y los PICs, sistema de alarmas, interfaz gráfica, etc. La participación del Departamento de Ingeniería de Sistemas Telemáticos (DIT) se centró en uno de los módulos, denominado Módulo de Usos y Demandas, situado funcionalmente como un elemento anexo al conjunto, que no afecta directamente al funcionamiento general, pero que puede aportar información muy valiosa sobre el sistema.

El objetivo de este Módulo de Usos y Demandas es bastante genérico y, a la vez, ambicioso. Se pretende que analice de un modo inteligente las consultas realizadas por los usuarios desde los PICs. De este análisis se pueden extraer conclusiones (conocimiento) sobre posibles patrones de comportamiento por parte de los usuarios, relaciones entre diferentes entidades (averías, fallos en las comunicaciones, consultas, momento del día, etc.) o cualquier otra información de interés, tanto para los suministradores de las bases de datos (Consorcio de Transportes de Madrid y Colegio de Farmacéuticos de Madrid) como para la empresa que se encargue del mantenimiento y explotación del sistema.

Desde un principio se confió en la posibilidad de aplicar el trabajo que se estaba desarrollando para esta Tesis, dentro del Módulo de Usos y Demandas del SEIC.

Por su propia naturaleza, para realizar pruebas exhaustivas en el Módulo de Usos y Demandas se requiere un volumen de datos grande que, por otro lado, no podrá recopilarse hasta que se construya el resto del sistema y se ponga en funcionamiento. Es decir, la calidad y relevancia de las reglas inducidas sólo se podrá valorar tras meses de pruebas de campo del sistema SEIC.

Por fortuna, el Consorcio Regional de Transportes de Madrid nos proporcionó desde el principio varios ficheros de datos, recopilados durante los meses de Julio a Noviembre de 1994 en diferentes puntos de Madrid, dentro del proyecto SIT (Sistema de Información de Transportes), precursor del actual SEIC. Estos datos representan diferentes características de las consultas que se realizaron en varios PICs, y con ellos se han podido realizar diferentes pruebas del sistema descrito en esta Tesis.

A continuación se describe el formato de los datos disponibles y los pasos de selección, preprocesado y transformación previos a la inducción de conocimiento (minería de datos), dentro del proceso global de KDD. También se presentan algunos de los resultados obtenidos alimentando el Módulo de Usos y Demandas con estos datos, así como algunas conclusiones derivadas de la interpretación y evaluación de los resultados.

5.3.1 Datos de entrada disponibles

El volumen de datos disponible (del proyecto SIT) corresponde a más de 40000 consultas, cada una de ellas caracterizada por 14 atributos, como se verá a continuación.

Estos ficheros de datos de entrada (de nombre XXXM.dat) vienen ordenados por el PIC en el que se han generado (XXX) y el mes (M) correspondiente. Existen datos de 8 diferentes localizaciones de PICs:

Tabla 5-1: Ubicación de los Puntos de Información Ciudadana

Nombre del PIC (XXX)	Localización del PIC
ALO	metro Alonso Martínez
ATC	metro Atocha Renfe
CAL	metro Callao
AVA	metro Avenida América
COR	El Corte Inglés de Princesa
CUA	metro Cuatro Caminos
PCA	metro Plaza Castilla
SOL	metro Sol

y de los meses de Julio a Noviembre de 1994:

Tabla 5-2: Meses en que se realizaron las consultas en el SIT

valor M	Mes
J	Julio
A	Agosto
S	Septiembre
O	Octubre
N	Noviembre

Dentro de cada fichero, los datos tienen un formato plano, correspondiendo cada línea a una única consulta. Dentro de cada línea, cada uno de los atributos se almacena en una columna fija, del siguiente modo:

Tabla 5-3: Formato de los ficheros históricos del SIT

Columna	Dato
1-2	día del mes (1-31)
3-4	mes (1-12)
5-8	año
9-10	día de semana (1-7) [1=domingo, ..., 7=sabado]
11-12	hora inicio de consulta (0-24)
13-14	minutos inicio consulta (0-59)
15-17	código de pantalla, especificar origen: P01 = desde aquí (in situ) P02 = calle o edificio singular P03 = cruce de calles P04 = estación de metro
18-21	duración de la consulta (segundos)
22-29	reservado
30-32	código de pantalla, especificar destino: P05 = hasta aquí P06 = calle o edificio singular P07 = cruce de calles P08 = estación de metro
33-44	reservado
45-47	código de pantalla, cambiar fecha/hora: P09 = no cambiar P10 = cambiar
48-59	reservado
60-62	código de pantalla, limitar modo: P11 = todos los modos P12 = sólo autobús P13 = sólo metro
63-74	reservado
75-77	código de pantalla, criterio de camino: P14 = camino óptimo (coste generalizado, tiempo ponderado y tarifa convertida en tiempo) P15 = minimizar transbordos P16 = minimizar tiempo
78-89	reservado

Para trabajar con estos datos iniciales del SIT es necesario preprocesarlos y transformarlos, para convertirlos a formato relacional, en diferentes tablas que puedan ser manejadas por el algoritmo de aprendizaje.

5.3.2 Selección y preprocesado

A partir de los ficheros de datos de partida, se pueden identificar los siguientes dominios, para los diferentes atributos:

Id_consulta: Entero que identifica de forma unívoca cada una de las consultas. Los posibles valores varían desde 1, para la primera consulta, hasta n (para n consultas almacenadas).

Cambio_fecha: Indica si se ha cambiado la fecha/hora actual para realizar la consulta. Los posibles valores son: {“SI”, “NO”}

Tipo_lugar: Identifica el origen o destino de una consulta. Puede tomar los valores: {“Aquí”, “Calle_Edif_singular”, “Cruce”, “Metro”}

Tipo_rest_tte: Restricción impuesta por el usuario respecto al medio de transporte que se debe emplear. Su dominio es: {“Cualquiera”, “Sólo_bus”, “Sólo_metro”}

Tipo_rest_optim: Restricción impuesta por el usuario respecto al criterio que se debe optimizar: {“Camino_óptimo”, “Min_transbordos”, “Min_tiempo”}

Localización: Lugar en que se encuentra un PIC (corresponde a la localización indicada por el fichero de entrada): {“Al_Martínez”, “Av_América”, “At_renfe”, “Callao”, “CI_Princesa”, “C_Caminos”, “P_Castilla”, “Sol”}

Mes: Mes del año en que se realizó la consulta (corresponde al mes indicado por el fichero de entrada correspondiente): {“Enero”, “Febrero”, “Marzo”, “Abril”, “Mayo”, “Junio”, “Julio”, “Agosto”, “Septiembre”, “Octubre”, “Noviembre”, “Diciembre”}

Día_semana: Día de la semana en que se realizó la consulta: {“Domingo”, “Lunes”, “Martes”, “Miércoles”, “Jueves” “Viernes”, “Sábado”}

Hora_día: Hora del día en que se hizo la consulta: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24}.

Duración: Indica la duración en segundos de la consulta. Sus valores varían de 60 en 60 segundos, desde 0 hasta 216000.

Se han suprimido algunos atributos existentes en los ficheros originales, dada su irrelevancia para nuestro módulo inductivo. Así, por ejemplo, se ha eliminado el año de cada consulta, ya que todas se realizaron en 1994.

En algunas de las pruebas realizadas se han introducido dominios borrosos nuevos, generados a partir de algunos de los dominios originales. En concreto, se ha introducido borrosidad en los valores de Hora_día y los de Duración, del siguiente modo (figura 5-1 y figura 5-2):

Hora_día_borrosa: Momento del día en que se hizo la consulta: {“Noche”, “Mañana”, “Tarde”}

Duración_borrosa: Indica la duración de la consulta. Sus valores son: {“Corta”, “Media”, “Larga”}

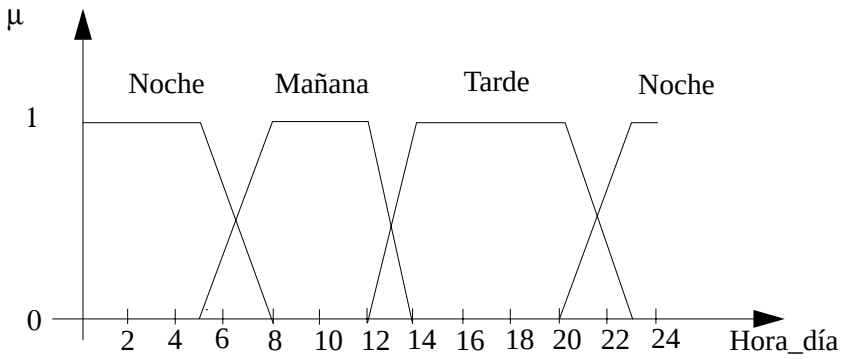

Figura 5-1: Subconjuntos borrosos para Hora_día

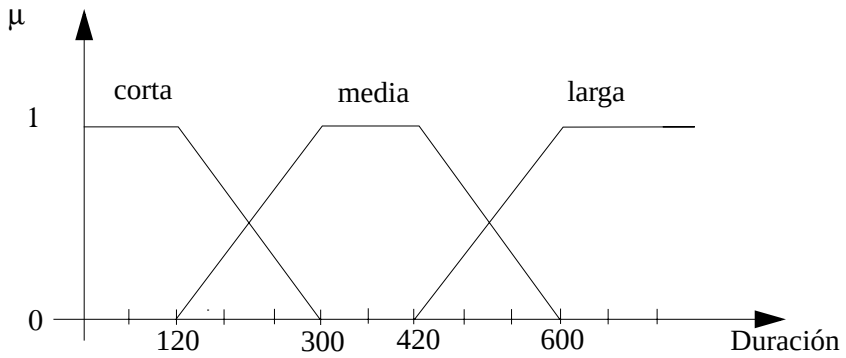

Figura 5-2: Subconjuntos borrosos para Duración

Dado el volumen de datos disponible, es conveniente aplicar técnicas de muestreo sobre los mismos. Para obtener los resultados mostrados en el apartado 5.3.4 se utilizó una muestra aleatoria con el 5% del total de consultas existentes. Esta muestra se mantuvo constante en todas las pruebas realizadas.

5.3.3 Transformación de los datos

Las relaciones construidas a partir de los datos iniciales son las siguientes (con los atributos indicados):

- Relación para describir el destino de cada una de las consultas:

  destino (Id_consulta, Tipo_lugar)

- Origen de cada consulta:

  origen (Id_consulta, Tipo_lugar)

- Otra fecha/hora para consulta:

  fecha (Id_consulta, Cambio_fecha)

- Restricciones de transporte de una consulta:

  rest_tte (Id_consulta, Tipo_rest_tte)

- Restricciones de optimización de una consulta:

  rest_optim (Id_consulta, Tipo_rest_optim)

- Lugar en que se ha realizado una consulta (ubicación de su PIC):

  realizada_en (Id_consulta, Localización)

- Momento en que se hizo una consulta, indicando mes, día de la semana y hora del día:

  cuando (Id_consulta, Mes, Día_semana, Hora_día)

- Duración de una consulta:

  duración (Id_consulta, Duración)

En las pruebas realizadas con datos borrosos se han introducido tres relaciones nuevas, una para cada uno de los valores borrosos definidos en el dominio *Hora_día_borrosa*:

- Mes y día de consultas realizadas por la mañana (construida con las tuplas de la relación *cuando* cuyo atributo borroso *Hora_día_borrosa* es igual a Mañana):

  cuando_mañana (Id_consulta, Mes, Día_semana)

  es decir,

  $$\text{cuando\_mañana}(X, Y, Z) \Leftrightarrow \text{cuando}(X, Y, Z, W) \wedge W = \text{Mañana}$$

- Mes y día de consultas realizadas por la tarde (construida con las tuplas de la relación *cuando* cuyo atributo borroso *Hora_día_borrosa* es igual a Tarde):

  cuando_tarde (Id_consulta, Mes, Día_semana)

  es decir,

  $$\text{cuando\_tarde}(X, Y, Z) \Leftrightarrow \text{cuando}(X, Y, Z, W) \wedge W = \text{Tarde}$$

- Mes y día de consultas realizadas por la noche (construida con las tuplas de la relación *cuando* cuyo atributo borroso *Hora_día_borrosa* es igual a Noche):

  cuando_noche (Id_consulta, Mes, Día_semana)

  es decir,

  $$\text{cuando\_noche}(X, Y, Z) \Leftrightarrow \text{cuando}(X, Y, Z, W) \wedge W = \text{Noche}$$

Del mismo modo, para los valores borrosos del dominio *Duración_borrosa* se han definido tres relaciones borrosas:

- Consultas cuya duración fue corta:

  duración_corta (Id_consulta)

- Consultas de duración media:

  duración_media (Id_consulta)

- Consultas largas:

  duración_larga (Id_consulta)

Mediante el uso de relaciones intensionales (apartado 4.3.5) se ofrece la posibilidad de que el operador introduzca conocimiento de base en el sistema. Estas relaciones intensionales permiten seleccionar el subconjunto de datos para el que se desea inducir una descripción lógica (en sí

mismas, también pueden considerarse como descripciones lógicas). Por ejemplo, para seleccionar las “consultas que tuvieron lugar en la estación de metro Sol durante el mes de Noviembre” se puede introducir la relación intensional:

I_sol_noviembre (a: Id_consulta) :-

realizada_en(a, b), =(b, Sol), cuando(a, c, d, e), =(c, Noviembre).

Estas relaciones intensionales se introducen en un fichero ascii, pudiendo coexistir varias. Al generarse la BD relacional (conjunto de entrenamiento para nuestro algoritmo), se añaden al final de la misma, detrás de las relaciones extensionales definidas anteriormente.

5.3.4 Resultados

A continuación se muestran diferentes resultados obtenidos a partir de los datos del SIT, con la versión actual de FZFOIL. Para la obtención de los mismos se ha utilizado una estación de trabajo SUN Ultra-1, con sistema operativo Solaris 2.5. El tiempo de ejecución máximo para cada ejemplo se ha fijado en 60 minutos (si en ese tiempo no se llega a una definición consistente y completa, se para la ejecución en el punto en que se encuentre y se muestran los resultados, sin aplicar ningún mecanismo de poda de definiciones).

En todos los ejemplos se ha utilizado la función *Interés** para evaluación de literales.

Se han seleccionado 6 relaciones objetivo diferentes, introducidas mediante relaciones intensionales[1]. Para cada una de ellas se han construido dos ejemplos, uno con relaciones ordinarias (caso a, en los siguientes apartados) y otro con relaciones ordinarias y borrosas (caso b), lo que hace un total de 12 ejemplos.

La selección, preprocesado y transformación de los datos de partida ha sido igual para todos los ejemplos, tal como se explica en los apartados anteriores. Las únicas diferencias entre ellos son, por tanto, la relación objetivo seleccionada y la existencia o no de relaciones borrosas (dependiendo de si se trata de un ejemplo de tipo b o del tipo a respectivamente).

Para cada ejemplo se muestra:

- Definición construida por FZFOIL después de 60 minutos de ejecución.
- LDI (expresada en bits) asociada a dicha definición.
- Valores de k-completitud y k-consistencia máximos permitidos para que dicha definición pueda considerarse k-completa y k-consistente.
- Valores de k-completitud y k-consistencia máximos por bit de LDI.

1. Consultas en las que se desea minimizar el tiempo

La relación objetivo utilizada para estos ejemplos corresponde a la siguiente relación intensional:

1. En el caso de seleccionar como relación objetivo una relación definida intensionalmente, es necesario realizar pequeñas modificaciones en el fichero de entrada, para evitar que FZFOIL induzca la misma definición utilizada para construir la relación intensional.
Una forma de conseguirlo es prohibir como “constantes de la teoría” (apartado C.3.1) todos los valores de los dominios utilizados en la construcción de la relación intensional objetivo.

```
I_min_tiempo(Id_consulta)
min_tiempo(X):-
        rest_optim(X,Y),
        =_const(Y,Min_tiempo).
```

a) Definición construida sólo con relaciones ordinarias.

El tiempo de ejecución finaliza durante la construcción de la tercera cláusula (C3), que no llega a ser consistente. Resulta, por tanto, una definición incompleta e inconsistente, que puede describirse como[1]:

"El 20.17% de las consultas en las que se quiere minimizar el tiempo del viaje cumplen:

- el identificador de la consulta coincide con la hora en que se realiza la consulta;
- o bien se realiza en Agosto, el identificador coincide con la hora de una segunda consulta realizada un Viernes de Octubre y existe una tercera que se realiza también en Octubre y el mismo día de la semana y hora que la primera;
- o bien, con un 61% de probabilidad, se realizan desde el metro de Sol para ir desde un origen que no es una estación de metro a un destino que tampoco es una estación de metro".

Las cláusulas de Horn construidas son:

```
[C1] {9.910 bits} [694.00/1694.00]→ [1.00/1.00]
min_tiempo ( A ) :-
        cuando ( A B C A ).
[C2] {61.163 bits} [693.00/1693.00]→ [4.00/4.00]
min_tiempo ( A ) :-
        cuando ( B C D A ),
        =_const ( C Octubre ),
        =_const ( D Viernes ),
        cuando ( E C F G ),
        cuando ( A H F G ),
        =_const ( H Agosto ).
[C3] {52.547 bits} [689.00/1689.00]→ [135.00/220.00]
min_tiempo ( A ) :-
        realizada_en ( A B ),
```

1. Hay que hacer notar que sobre estas definiciones parcialmente construidas no se ha aplicado ningún mecanismo de poda de descripciones, ni ningún mecanismo para eliminar literales redundantes, por lo que no pueden considerarse como los resultados finales del FZFOIL. El objetivo de mostrar estos resultados parciales (después de 60 minutos de ejecución) es el de comparar el proceso de aprendizaje con y sin relaciones borrosas.
Si se ejecutase el algoritmo hasta el final, se aplicarían los mecanismos de poda y, posiblemente, el resultado sería que no existe ninguna definición suficientemente completa y consistente para muchas de estas relaciones objetivo.

=_const (B Sol),
origen (A C),
¬ =_const (C Metro),
origen (D C),
destino (A E),
¬ =_const (E Metro),
destino (D F)...

Los valores de los parámetros a comparar son los siguientes:

LDI = 121.035 bits

k-completitud = 140 / 694 = 20.17% k-completitud / LDI = 0.0017

k-consistencia = 140 / 225 = 62.22% k-consistencia / LDI = 0.0051

b) Definición construida con relaciones borrosas:

La definición parcialmente construida consta, en este caso de una única cláusula, inconsistente e incompleta, que puede interpretarse como:

"El 49.90% de las consultas en las que se quiere minimizar el tiempo del viaje cumplen que, con un 44.81% de probabilidad, se realizaron por la tarde un día que no era Viernes".

[C1] {17.062 bits} [694.00/1694.00]→ [346.28/772.71]
min_tiempo (A) :-
cuando_tarde (A B C),
¬ =_const (C Viernes)...

LDI = 17.062 bits

k-completitud = 346.28 / 694 = 49.90% k-completitud / LDI = 0.0293

k-consistencia = 346.28 / 772.71 = 44.81% k-consistencia / LDI = 0.0263

2. Consultas en las que se desea minimizar el número de transbordos

Relación intensional objetivo:

I_min_transbordos(Id_consulta)
min_transbordos(X):-
rest_optim(X,Y),
=_const(Y,Min_transbordos).

a) Definición construida con relaciones ordinarias:

La definición parcialmente construida consta, en este caso, de cuatro cláusulas, que pueden interpretarse del siguiente modo:

"El 58.45% de las consultas en las que se quiere minimizar el número de transbordos cumplen:

- no se realizan ni en el metro de Alonso Martínez, ni en el Corte Inglés de Princesa, ni en el metro de Avenida de América, y el origen de la consulta es una estación de Metro;
- o bien, no se realizan ni en el metro de Sol ni en el de Cuatro Caminos y el origen de la consulta es una estación de metro;
- o bien, no se realizan ni en el metro de Alonso Martínez, ni en el Corte Inglés de Princesa, ni en el metro de Avenida de América, ni en el de Sol, el origen es una calle o edificio singular y es distinto de su destino;
- o bien, con un 51% de probabilidad, no se realizan ni en el metro de Alonso Martínez, ni en el Corte Inglés de Princesa, ni en el metro de Avenida de América, el origen no es un cruce de calles y el destino es distinto del origen":

[C1] {36.203 bits} [1119.00/2049.00]→ [217.00/217.00]

```
min_transbordos ( A ) :-
        realizada_en ( A B ),
        ¬ =_const ( B Al_Martínez ),
        ¬ =_const ( B CI_Princesa ),
        ¬ =_const ( B Av_América ),
        origen ( A C ),
        =_const ( C Metro ).
```

[C2] {31.087 bits} [902.00/1832.00]→ [88.00/88.00]

```
min_transbordos ( A ) :-
        realizada_en ( A B ),
        ¬ =_const ( B Sol ),
        ¬ =_const ( B C_Caminos ),
        origen ( A C ),
        =_const ( C Metro ).
```

[C3] {46.703 bits} [814.00/1744.00]→ [50.00/52.00]

```
min_transbordos ( A ) :-
        realizada_en ( A B ),
        ¬ =_const ( B Al_Martínez ),
        ¬ =_const ( B CI_Princesa ),
        ¬ =_const ( B Av_América ),
        ¬ =_const ( B Sol ),
        origen ( A C ),
        ¬ destino ( A C ),
        =_const ( C Calle_Edif_singular ).
```

[C4] {42.003 bits} [764.00/1692.00]→ [299.00/542.00]

min_transbordos (A) :-

realizada_en (A B),

¬ =_const (B Al_Martínez),

¬ =_const (B CI_Princesa),

¬ =_const (B Av_América),

origen (A C),

¬ destino (A C),

¬ =_const (C Cruce)...

LDI = 151.411 bits

k-completitud = 654 / 1119 = 58.45% k-completitud / LDI =0.0039

k-consistencia = 654 / 899 = 72.75% k-consistencia / LDI = 0.0048

b) Definición construida con relaciones borrosas:

De nuevo resulta una definición más sencilla para el caso de relaciones borrosas:

"El 24.43% de las consultas en las que se quiere minimizar el número de transbordos se realizan (con una probabilidad del 62.54%) por la mañana, un día que no es Martes ni en Octubre ni en Julio".

[C1] {31.291 bits} [1119.00/2049.00]→ [273.35/437.05]

min_transbordos (A) :-

cuando_mañana (A B C),

¬ =_const (B Octubre),

¬ =_const (C Martes),

¬ =_const (B Julio)...

LDI = 31.291 bits

k-completitud = 273.35 / 1119 = 24.43% k-completitud / LDI = 0.0078

k-consistencia = 273.35 / 437.05 = 62.54% k-consistencia / LDI = 0.0200

3. Consultas en las que se desea usar sólo el autobús

Relación intensional objetivo:

I_sólo_bus(Id_consulta)

sólo_bus(X):-

rest_tte(X,Y),

=_const(Y,Sólo_bus).

a) Definición construida con relaciones ordinarias:

[C1] {38.016 bits} [232.00/1232.00]→ [6.00/12.00]
sólo_bus (A) :-
realizada_en (A B),
=_const (B CI_Princesa),
origen (A C),
¬ =_const (C Calle_Edif_singular),
¬ =_const (C Aquí),
¬ destino (A C)...

LDI = 38.016 bits
k-completitud = 6 / 232 = 2.59% k-completitud / LDI = 0.0007
k-consistencia = 6 / 12 = 50.00% k-consistencia / LDI = 0.0131

b) Definición construida con relaciones borrosas:

[C1] {31.291 bits} [232.00/1232.00]→ [92.83/405.23]
sólo_bus (A) :-
cuando_tarde (A B C),
¬ =_const (B Noviembre),
¬ =_const (C Domingo),
¬ =_const (C Miércoles)...

LDI = 31.291 bits
k-completitud = 92.83 / 232 = 40.01% k-completitud / LDI = 0.0128
k-consistencia = 92.83 / 405.23 = 22.91% k-consistencia / LDI = 0.0073

4. Consultas realizadas en Alonso Martínez, en las que se desea minimizar el tiempo del viaje

Relación intensional objetivo:

```
I_alonso_mintiempo(Id_consulta)
alonso_mintiempo(X):-
    realizada_en(X, Y),
    =_const(Y, Al_Martínez),
    rest_optim(X, Z),
    =_const(Z, Min_tiempo).
```

a) Definición construida con relaciones ordinarias:

[C1] {18.195 bits} [70.00/1070.00]→ [8.00/13.00]
alonso_mintiempo (A) :-

cuando (B C D A),
=_const (D Lunes)...

LDI = 18.195 bits

k-completitud = 8 / 70 = 11.43% k-completitud / LDI = 0.0063

k-consistencia = 8 / 13 = 61.54% k-consistencia / LDI = 0.0338

b) Definición construida con relaciones borrosas:

[C1] {65.925 bits} [70.00/1070.00]→ [38.34/331.05]

alonso_mintiempo (A) :-

cuando (A B C D),
¬ =_const (B Julio),
¬ cuando_noche (A B C),
¬ =_const (C Martes),
¬ =_const (C Lunes),
¬ =_const (B Agosto),
¬ =_const (C Domingo),
¬ =_const (C Jueves)...

LDI = 65.925 bits

k-completitud = 38.34 / 70 = 54.77% k-completitud / LDI = 0.0083

k-consistencia = 38.34 / 331.05 = 11.58% k-consistencia / LDI = 0.0018

5. Consultas en las que se desea minimizar el número de transbordos, usando sólo el autobús.

Relaciones intensionales[1]:

```
I_bus_mintransbordos(Id_consulta)
bus_mintransbordos(X):-
    rest_tte(X, Y),
    =_const(Y, Sólo_bus),
    rest_optim(X, Z),
    =_const(Z, Min_transbordos).
*I_sólo_bus(Id_consulta)
sólo_bus(X):-
    rest_tte(X,Y),
```

1. En este ejemplo se utilizan tres relaciones intensionales pero sólo la primera es relación objetivo. Por tanto, para evitar definiciones infructuosas (tal como se indica en la nota 1, a pie de pág. 135), basta con prohibir como constantes de la teoría los valores de los dominios utilizados en la construcción de dicha relación objetivo intensional: Id_consulta, Tipo_rest_tte y Tipo_rest_optim.

```
        =_const(Y,Solo_bus).
*I_min_transbordos(Id_consulta)
min_transbordos(X):-
        rest_optim(X,Y),
        =_const(Y,Min_transbordos).
```

a) Definición construida con relaciones ordinarias.

En este caso se obtiene una definición consistente y completa, formada por una única cláusula, que utiliza una de las relaciones intensionales como antecedente: “todas las consultas realizadas en las que se desea utilizar sólo el autobús, quieren también minimizar el número de transbordos”.

```
[C1] {4.907 bits} [232.00/1232.00]→ [232.00/232.00]
bus_mintransbordos ( A ) :-
        sólo_bus ( A ).
```

LDI = 4.907 bits

k-completitud = 232 / 232 = 100% k-completitud / LDI = 0.2038

k-consistencia = 232 / 232 = 100% k-consistencia / LDI = 0.2038

b) Definición construida con relaciones borrosas.

La definición inducida en el caso de tener relaciones borrosas es la misma que en el caso de sólo relaciones ordinarias, ya que se obtiene una definición consistente y completa sin necesidad de utilizar literales borrosos:

```
[C1] {5.392 bits} [232.00/1232.00]→ [232.00/232.00]
bus_mintransbordos ( A ) :-
        sólo_bus ( A ).
```

La LDI calculada para la definición en este caso borroso es mayor que la calculada en el caso a, aunque ambas definiciones son iguales. Esto se debe a que la longitud de codificación de los literales depende del número de relaciones en la BD (apartado 4.3.2.9), y en los ejemplos de tipo b se están considerando 6 relaciones borrosas más que en los ejemplos de tipo a (sin relaciones borrosas).

LDI = 5.392 bits

k-completitud = 232 / 232 = 100% k-completitud / LDI = 0.1854

k-consistencia = 232 / 232 = 100% k-consistencia / LDI = 0.1854

6. Consultas referentes al viernes por la tarde:

Relación intensional objetivo usada con relaciones ordinarias:

```
I_viernes_tarde(Id_consulta)
viernes_tarde(X):-
        cuando(X, Y, Z, W),
```

=_const(Z, Viernes),
>_const(W, 13),
¬ >_const(W, 21).

Relación intensional objetivo usada con relaciones borrosas:

I_viernes_tarde(Id_consulta)
viernes_tarde(a):-
cuando_tarde(a, b, c),
=_const(c, Viernes).

a) Definición construida con relaciones ordinarias:

[C1] {53.234 bits} [172.00/1166.00]→ [2.00/2.00]
viernes_tarde (A) :-
cuando (B C D A),
cuando (A C E F),
cuando (F G E H),
cuando (H C I J).
[C2] {61.952 bits} [170.00/1164.00]→ [50.00/196.00]
viernes_tarde (A) :-
realizada_en (A B),
¬ =_const (B At_renfe),
¬ =_const (B P_Castilla),
origen (A C),
¬ =_const (C Cruce),
origen (D C),
¬ realizada_en (D B),
destino (A C),
¬ =_const (B Av_América),
¬ =_const (B Callao)...

LDI = 114.186 bits
k-completitud = 52 / 172 = 30.23% k-completitud / LDI =0.0026
k-consistencia = 52 / 198 = 26.26% k-consistencia / LDI = 0.0023

b) Definición construida con relaciones borrosas:

[C1] {9.155 bits} [166.66/1203.00]→ [166.66/678.59]
viernes_tarde (A) :-
cuando_tarde (A B C)...

LDI = 9.155 bits

k-completitud = 166.66 / 166.66 = 100% k-completitud / LDI =0.1095

k-consistencia = 166.66 / 678.59 = 24.56% k-consistencia / LDI = 0.0269

5.3.5 Comparación de resultados

La comparación la realizaremos a partir de los cinco parámetros calculados para cada ejemplo:

- LDI de la definición construida
- Valor de k-completitud máximo permitido para que dicha definición pueda considerarse k-completa.
- Valor de k-consistencia máximo permitido para que dicha definición pueda considerarse k-consistente.
- Valores de k-completitud máximo por unidad de longitud intensional.
- Valores de k-consistencia máximo por unidad de longitud intensional.

Tabla 5-4: Comparación de LDI para diferentes ejemplos

	caso A (relaciones ordinarias)	**caso B (relaciones ordinarias y borrosas)**	**Relación: LDI(A) / LDI(B)**
Ejemplo 1	121.035 bits	**17.062 bits**	7.1
Ejemplo 2	151.411 bits	**31.291 bits**	4.8
Ejemplo 3	38.016 bits	**31.291 bits**	1.2
Ejemplo 4	**18.195 bits**	65.925 bits	0.3
Ejemplo 5	**4.907 bits**	5.392 bits	0.9
Ejemplo 6	114.186 bits	**9.155 bits**	12.5

Puede observarse que, para este conjunto de ejemplos, la longitud de descripción intensional tiende a ser mayor en el caso de sólo relaciones ordinarias (hasta 12.5 veces mayor en el ejemplo 6), aunque en algunos casos ocurre lo contrario[1] (ejemplo 4). En media, con las definiciones borrosas se ha conseguido un ahorro de más del 60% del total de bits.

1. El ejemplo número 5 tiene una LDI algo mayor para el caso borroso, aunque, como ya se ha dicho, no se debe a que la definición borrosa que se construye sea más compleja (realmente es la misma), sino a que cambia la BD de entrada (se consideran 6 relaciones borrosas más).
Esta penalización en la LDI debida al número de relaciones de la BD se está aplicando en todos los ejemplos de tipo b (todos tienen 6 relaciones más que sus equivalentes de tipo a).

Esto se traduce en descripciones borrosas más cortas y, por tanto, más fáciles de entender.

Tabla 5-5: Comparación del valor de k-completitud

	caso A (relaciones ordinarias)	**caso B (relaciones ordinarias y borrosas)**	**Relación: k-compl(A) / k-compl(B)**
Ejemplo 1	0.2017	**0.4990**	0.40
Ejemplo 2	**0.5845**	0.2443	2.39
Ejemplo 3	0.0259	**0.4001**	0.06
Ejemplo 4	0.1143	**0.5477**	0.21
Ejemplo 5	**1.0000**	**1.0000**	1.00
Ejemplo 6	0.3023	**1.0000**	0.30

El máximo valor de k-completitud (para el que se satisface la condición de k-completitud) es, normalmente mayor en el caso B (definiciones borrosas), aunque con algunas excepciones (ejemplo 2). Esto puede interpretarse como que las descripciones borrosas son más generales que las construidas sólo con relaciones ordinarias.

Tabla 5-6: Comparación del valor de k-consistencia

	caso A (relaciones ordinarias)	**caso B (relaciones ordinarias y borrosas)**	**Relación: k-consis(A) / k-consis(B)**
Ejemplo 1	**0.6220**	0.4481	1.39
Ejemplo 2	**0.7275**	0.6254	1.16
Ejemplo 3	**0.5000**	0.2291	2.18
Ejemplo 4	**0.6154**	0.1158	5.31
Ejemplo 5	**1.0000**	**1.0000**	1.00
Ejemplo 6	**0.2626**	0.2456	1.07

El máximo valor de k-consistencia (para el que se satisface la condición de k-consistencia) es, en los ejemplos anteriores, siempre mayor para el caso A (sólo relaciones ordinarias). Esto se

traduce en unas definiciones borrosas menos precisas que las construidas sólo con relaciones ordinarias.

Tabla 5-7: Comparación de valores de k-completitud y k-consistencia por unidad de LDI

	k-completitud / LDI		**k-consistencia / LDI**	
	caso A (relaciones ordinarias)	**caso B (relaciones ordinarias y borrosas)**	**caso A (relaciones ordinarias)**	**caso B (relaciones ordinarias y borrosas)**
Ejemplo 1	0.0017	**0.0293**	0.0051	**0.0263**
Ejemplo 2	0.0039	**0.0078**	0.0048	**0.0200**
Ejemplo 3	0.0007	**0.0128**	**0.0132**	0.0073
Ejemplo 4	0.0063	**0.0083**	**0.0338**	0.0018
Ejemplo 5	**0.2038**	0.1854	**0.2038**	0.1854
Ejemplo 6	0.0026	**0.1095**	0.0023	**0.0269**

Al comparar los valores de k-completitud y k-consistencia por unidad de LDI, se observa que, en los ejemplos considerados:

- el valor k-completitud/LDI es mayor en descripciones borrosas[1]
- el valor k-consistencia/LDI no siempre es mayor en descripciones ordinarias.

Este resultado puede interpretarse como que, para longitudes de descripción fijas, las definiciones borrosas son más completas y, algunas veces, más consistentes que las definiciones ordinarias. O dicho de otro modo, las definiciones borrosas generalizan mejor los conceptos y, en ocasiones, son también más precisas que las definiciones sin relaciones borrosas.

5.4 Análisis de la complejidad

La complejidad asociada a la inducción de una definición lógica viene determinada por la búsqueda de los literales que constituyen sus cláusulas. A su vez, esta búsqueda depende de:

- Tamaño del espacio de búsqueda, formado por todos los literales candidatos a ser añadidos a una cláusula en construcción.
- Tamaño de los conjuntos intermedios de tuplas, con los que se evalúa cada literal candidato.

Ambos valores vienen determinados por factores como el grado de las relaciones existentes en la BD, el predicado elegido como objetivo o la longitud de la cláusula resultante.

1. Con la excepción del ejemplo 5, tal como se explica en la nota a pie de pág. 144

Para realizar un estudio formal de la complejidad de FOIL, utilizaremos el enfoque seguido por [Pazzani y Kibler, 92], en el que descomponen el coste de la búsqueda de un nuevo antecedente L_i en una cláusula en construcción C^{i-1} con *u* variables, en el producto de dos parámetros:

- *Coste teórico* (CT), que indica el número de literales diferentes que pueden ser añadidos a la cláusula (es decir, mide el tamaño del espacio de búsqueda).
- *Coste de evaluación de un literal* (CE), que mide el coste asociado a la evaluación de un literal candidato, con las tuplas del conjunto intermedio correspondiente. Este valor viene determinado por el número de tuplas usadas para la evaluación de cada antecedente.

El coste resultante para la búsqueda del antecedente i-ésimo L_i, en una cláusula, será el producto de ambos parámetros:

$$\text{Coste}(L_i) = CT \cdot CE \qquad \text{[EC. 5.1]}$$

Para estudiar la complejidad del nuevo sistema FZFOIL seguiremos un proceso paralelo, en el que se calcularán los correspondientes valores CT y CE, y se compararán con los obtenidos para FOIL.

5.4.1 Complejidad de FOIL

5.4.1.1 Coste Teórico

Como ya se ha definido, el coste teórico (CT) mide el tamaño del espacio de búsqueda, expresado en número de literales.

Para cada literal que se añade a una cláusula existe un espacio de búsqueda diferente. Supongamos que queremos añadir un nuevo antecedente L_i a una cláusula en la que existen *u* variables diferentes. ¿Cuántos candidatos deberemos evaluar hasta elegir L_i?

- Para cada predicado q_m, asociado a una relación Q_m de la BD, se generarán, en el espacio de búsqueda, cierto número de literales afirmativos diferentes (dependiendo del grado *m* de Q_m y del número *u* de variables existentes en la cláusula) que denominaremos:

  NumLit(m,u) = número de literales afirmativos que existen por cada predicado q_m de grado *m* de la BD, candidatos a ser antecedentes de una cláusula en construcción con *u* variables usadas.

- Si en la BD existen

  NumPr(m) = número de predicados de grado *m* (un predicado para cada relación de grado *m* de la BD)

  y denominamos

  MaxG = máximo grado de las relaciones de la BD

  podemos obtener ya una expresión para el tamaño del espacio de búsqueda o coste teórico:

$$CT = 2 \cdot \sum_{i=1}^{MaxG} \text{NumPr}(i) \cdot \text{NumLit}(i, u) \qquad \text{[EC. 5.2]}$$

  El factor 2 de esta expresión se debe a que por cada literal afirmativo se considera otro literal negativo.

En la tabla 5-8 se muestran algunos valores de *NumLit(m,u)* que, como se ve, crecen rápidamente tanto con el grado *m* del predicado candidato como con el número *u* de variables usadas en la cláusula.

Tabla 5-8: Algunos valores de NumLit(m,u)

		u						
		1	**2**	**3**	**4**	**5**	**6**	**7**
m	**1**	1	2	3	4	5	6	7
	2	3	8	15	24	35	48	63
	3	10	32	72	136	230	360	532
	4	37	136	357	784	1525	2712	4501
	5	151	622	1863	4684	10375	20826	38647
	6	674	3060	10278	29168	72810	163764	338030

La expresión exacta de *NumLit(m,u)* no es sencilla (véase apartado B.2), pero se puede acotar fácilmente por su valor máximo, para cualquier predicado de la BD, durante la búsqueda de cualquier antecedente de la cláusula. Si denominamos

TotalPr	al número de relaciones de la BD
MaxG	al máximo grado de las relaciones existentes
MaxU	al número máximo de variables diferentes que pueden ser usadas en cualquier cláusula en construcción

tendremos, en el caso peor, que todos los predicados de la BD son de grado *MaxG* y que la cláusula en construcción tienen *MaxU* variables diferentes. En este caso el número de literales afirmativos para cada relación sería

$$\text{NumLit}(\text{MaxG}, \text{MaxU})$$

y por ello

$$CT_{max} = 2 \cdot \text{TotalPr} \cdot \text{NumLit}(\text{MaxG}, \text{MaxU}) \qquad \text{[EC. 5.3]}$$

Como todos los candidatos deben tener, al menos, una variable usada, un literal de grado máximo podrá usar, como mucho, *MaxG-1* variables nuevas, por lo que los diferentes literales generados para un predicado de grado *MaxG* serán variaciones de *MaxG* variables elegidas entre *MaxU+MaxG-1* posibles (aunque no todas las variaciones de variables nuevas son diferentes):

$$\text{NumLit}(\text{MaxG}, \text{MaxU}) \leq (\text{MaxU} + \text{MaxG} - 1)^{\text{MaxG}} \qquad \text{[EC. 5.4]}$$

De este modo, se puede obtener una cota superior para el valor de CT asociado a la búsqueda de un literal L_i:

$$CT \leq 2 \cdot \text{TotalPr} \cdot (\text{MaxU} + \text{MaxG} - 1)^{\text{MaxG}} \qquad \text{[EC. 5.5]}$$

Se pueden deducir varias conclusiones de esta expresión:

- El CT máximo crece aproximadamente de forma lineal con el número de predicados de la BD.
- El CT máximo crece polinómicamente con el número de variables usadas en la cláusula.
- El CT máximo crece exponencialmente con el grado de los predicados de la BD.

El mayor valor de CT se alcanza durante la construcción de las cláusulas más largas de una definición (midiendo la longitud por el número de variables que utiliza), especialmente cuando los literales candidatos usan predicados de grado alto. Por ejemplo, en una BD con los predicados *lista*, *vacía* y *componentes*, cuyos grados son 1, 1 y 3 respectivamente, los valores de CT durante la construcción de la regla:

lista(X_1) :- vacía(X_1)

lista(X_1) :- componentes(X_1, V_2, V_3), lista(V_3)

son (según [EC. 5.2] y tabla 5-8):

- para el primer literal de cualquiera de las dos cláusulas:

  $CT = 2 \cdot (2 \cdot 1 + 1 \cdot 10) = 24$ literales evaluados

- para el segundo literal (*lista(V_3)*) de la segunda cláusula:

  $CT = 2 \cdot (2 \cdot 3 + 1 \cdot 72) = 156$ literales evaluados

Por tanto, el CT asociado a la construcción de una definición D (número de literales evaluados durante la construcción de sus cláusulas) se puede aproximar por el CT asociado a la búsqueda del último literal de la cláusula más larga, suponiendo que los únicos predicados de la BD son los de mayor grado:

$$CT_D \approx 2 \cdot NumPr(MaxG) \cdot NumLit(MaxG, MaxV) \qquad \text{[EC. 5.6]}$$

$$CT_D \leq 2 \cdot NumPr(MaxG) \cdot (MaxV + MaxG - 1)^{MaxG} \qquad \text{[EC. 5.7]}$$

siendo

NumPr(MaxG)	el número de predicados de la BD con grado *MaxG* (máximo)
MaxV	el mayor número de variables distintas que existen en alguna de las cláusulas de la definición, excluyendo las introducidas por el último antecedente

En el ejemplo anterior, la aproximación [EC. 5.6] daría $CT_D \approx 2 \cdot 72 = 144$ literales, mientras que el valor real es $CT_D = 24 + 24 + 156 = 204$.

5.4.1.2 Coste de Evaluación

Una vez obtenido el tamaño del espacio de búsqueda de literales, se puede calcular el Coste de Evaluación (CE), asociado a la evaluación de cada literal candidato. El CE dependerá del tamaño N_i del conjunto intermedio de tuplas T_i que se considera para la selección del literal L_i de una cláusula.

El tamaño de cada conjunto intermedio de tuplas, para la búsqueda de los sucesivos literales antecedentes de una cláusula, depende de los literales seleccionados anteriormente (del predicado con el que se construyen y del número de variables nuevas que introducen). Para estimar el tamaño N_i de estos conjuntos T_i, [Pazzani y Kibler, 92] definen algunos conceptos:

- Densidad de un predicado q: porcentaje de casos en los que el predicado se satisface

$$\text{densidad}(q) = \frac{P}{P + N} \leq 1 \qquad \text{[EC. 5.8]}$$

siendo

P el número de tuplas que pertenecen a la relación Q

N el número de tuplas que no pertenecen a Q

Por ejemplo, para el predicado *menor_que*, definido sobre los pares de valores <a, b>, donde a y b pertenecen al dominio de los enteros del 1 al 10, valdrá *densidad(menor_que) = 45/100 = 0.45*

Para un literal $L_i = q(V_1, ..., V_m)$ que no introduzca variables nuevas, el tamaño del conjunto T_{i+1} se podrá estimar como:

$$N_{i+1} \approx N_i \cdot \text{densidad}(q) \qquad \text{[EC. 5.9]}$$

- Potencia de un predicado: máximo número de tuplas que lo satisfacen cuando uno de sus atributos es constante y los demás variables.

Por ejemplo, para el predicado *menor_que* definido anteriormente, este valor será *potencia(menor_que) = 9*, y se alcanza para los literales: *menor_que(1, V_2)* y *menor_que(V_1, 10)*.

La potencia de un predicado q permite acotar el tamaño del nuevo T_{i+1} generado por un literal $L_i=q(V_1, ..., V_m)$ que introduce nuevas variables:

$$N_{i+1} \leq N_i \cdot \text{potencia}(q) \qquad \text{[EC. 5.10]}$$

- Potencia media de un predicado: número medio de tuplas que lo satisfacen cuando uno de sus atributos es constante y los demás son variables.

Por ejemplo, para el predicado *menor_que* definido anteriormente, la potencia media será *potencia_med(menor_que) = 45/10 = 4.5* (aunque *menor_que(1, V_2)* se satisface en 9 casos y *menor_que(10, V_2)* nunca).

Se puede estimar el tamaño N_{i+1} del conjunto T_{i+1}, generado por un literal $L_i=q(V_1,..., V_m)$ que introduzca nuevas variables:

$$N_{i+1} \approx N_i \cdot \text{potencia\_med}(q) \qquad \text{[EC. 5.11]}$$

- Crecimiento máximo de un literal $L_i=q(V_1,..., V_m)$: indica el máximo valor posible de la fracción N_{i+1}/N_i. Vendrá dado por el máximo valor posible de densidad (es decir, la unidad) cuando L_i no introduzca variables nuevas, y por la potencia en caso contrario:

$$\text{crec\_máx}(L_i) = \begin{cases} 1, & \text{si } V_1, \ldots, V_m \text{ son usadas} \\ \text{potencia}(q), & \text{si alguna } V_1, \ldots, V_m \text{ es nueva} \end{cases} \qquad \text{[EC. 5.12]}$$

- Crecimiento medio de un literal $L_i=q(V_1,..., V_m)$: indica el valor medio de N_{i+1}/N_i. Dependerá de si L_i utiliza sólo variables usadas o si introduce alguna variable nueva:

$$\text{crec\_med}(L_i) = \begin{cases} \text{densidad}(q), & \text{si } V_1, \ldots, V_m \text{ usadas} \\ \text{potencia\_med}(q), & \text{si alguna } V_1, \ldots, V_m \text{ nueva} \end{cases} \qquad \text{[EC. 5.13]}$$

Haciendo uso de las anteriores definiciones, podremos estimar el tamaño del conjunto de entrenamiento T_i (usado para la búsqueda de L_i en una cláusula C^{i-1}) como:

$$CE = N_i \approx N_1 \cdot \prod_{j=1}^{i-1} crec\_med(L_j) \qquad \text{[EC. 5.14]}$$

y, en el caso peor, nunca superará:

$$N_i \leq N_1 \cdot \prod_{j=1}^{i-1} crec\_max(L_j) \qquad \text{[EC. 5.15]}$$

5.4.1.3 Conclusiones

El tamaño del espacio de búsqueda de literales determina el coste teórico, asociado a la búsqueda de un literal. Este espacio de búsqueda crece del siguiente modo:

- Linealmente respecto del número de predicados de la BD.
- Polinómicamente con el número de variables usadas en la cláusula.
- Exponencialmente con el grado de los predicados de la BD.

El tamaño del conjunto de entrenamiento también puede crecer durante la construcción de una cláusula, pero su tamaño máximo vendrá dado por la potencia de los literales que introducen nuevas variables.

El coste asociado a la evaluación de todos los literales candidatos dependerá del tamaño del espacio de búsqueda y del tamaño del conjunto de entrenamiento.

En algunos casos, especialmente durante la construcción de cláusulas con muchas variables distintas, puede ser necesario explorar espacios de búsqueda muy grandes, resultando muy costosa una búsqueda exhaustiva. Aunque en FOIL se aplican algunos heurísticos de poda (apartado 2.5.4.3), muchas veces no son suficientes para reducir la búsqueda de forma notable, por lo que resulta de interés el estudio de otros heurísticos de búsqueda más eficientes (apartado 6.2.1.2).

5.4.2 Complejidad de FZFOIL

Para comparar la complejidad del algoritmo FZFOIL respecto de FOIL, nos fijaremos en los dos parámetros con que se midió la complejidad en FOIL:

- Coste teórico, debido al tamaño del espacio de búsqueda de literales.
- Coste de evaluación, debido al tamaño del conjunto de tuplas con que se evalúa cada literal.

5.4.2.1 Coste Teórico

Para calcular el coste teórico de FZFOIL, hay que distinguir entre literales ordinarios y borrosos:

- Para cada predicado ordinario se generan los mismos literales que en FOIL, resultantes de las diferentes combinaciones de variables nuevas y usadas en sus atributos y de la utilización o no de la negación lógica. Por tanto, el coste teórico asociado a la búsqueda de literales ordinarios es el calculado por la expresión [EC. 5.2].
- Para los predicados borrosos es posible construir literales a través de la combinación de variables nuevas y usadas en sus atributos, así como aplicando diferentes etiquetas lingüísti-

cas a cada uno de los literales resultantes. Por tanto, el coste teórico asociado a los literales borrosos es mayor que en el caso de los literales ordinarios (para cada literal sin etiqueta existen tantos literales borrosos como etiquetas lingüísticas se permitan).

Por tanto, la expresión para el coste teórico en FZFOIL es la siguiente:

$$CT = 2 \cdot \sum_{i=1}^{MaxG} NumPrOrdinarios(i) \cdot NumLit(i, u) + \quad \text{[EC. 5.16]}$$

$$+ NumEtiquetas \cdot \sum_{i=1}^{MaxG} NumPrBorrosos(i) \cdot NumLit(i, u)$$

siendo

NumPrOrdinarios(m) = número de predicados ordinarios de grado *m* (un predicado para cada relación ordinaria de grado *m* de la BD)

NumPrBorrosos(m) = número de predicados borrosos de grado *m* (un predicado para cada relación borrosa de grado *m* de la BD)

NumLit(m,u) = número de literales sin calificador de verdad (sin negar y sin etiqueta lingüística) que pueden construirse por cada predicado de grado *m* de la BD, con un máximo de *u* variables usadas (véase apartado B.2)

MaxG = máximo grado de las relaciones de la BD

NumEtiquetas = número de etiquetas lingüísticas a considerar para cada literal borroso.

Si denominamos CT_{FOIL} al valor de CT obtenido en la expresión [EC. 5.2], y CT_{FZFOIL} al valor de CT obtenido para FZFOIL ([EC. 5.16]), se puede encontrar la siguiente relación entre ambas:

$$CT_{FZFOIL} = CT_{FOIL} + \quad \text{[EC. 5.17]}$$

$$+ (NumEtiquetas - 2) \cdot \sum_{i=1}^{MaxG} NumPrBorrosos(i) \cdot NumLit(i, u)$$

Como el número de etiquetas lingüísticas (NumEtiquetas) que se consideran para cada literal borroso suele ser mayor que dos (el valor 2 correspondería a un literal afirmativo y a su negado), entonces, resulta que el coste teórico es superior en FZFOIL que en FOIL.

5.4.2.2 Coste de Evaluación

La utilización de conjuntos proyectados de tuplas, tal como se hace con el heurístico *Interés** en FZFOIL. permite, en algunos casos, no tener que evaluar cada literal L_i con todas las tuplas del conjunto T_i. Esto es así porque los parámetros de evaluación se miden en el conjunto proyectado $T_1^{[i]}$, en el que cada tupla puede tener varias extensiones en T_i. Si cualquiera de las extensiones de una tupla de $T_1^{[i]}$ satisface un literal candidato L_i, entonces no será necesario evaluar con el resto de extensiones de la misma tupla. Esta idea es aplicable sólo para literales ordinarios aplicados sobre conjuntos de tuplas no borrosas, pero no para literales y/o conjuntos de tuplas borrosas (la existencia de tuplas y/o relaciones borrosas obliga a evaluar con todas las tuplas, por lo que el CE es el indicado en la [EC. 5.14]).

Por cada tupla de T_1 existirán, en media,

$$\prod_{j=1}^{i-1} crec\_med(L_j)$$

tuplas expandidas en T_i, tal como se deduce de la expresión [EC. 5.14].

En un conjunto de N_i tuplas, el número medio de tuplas que satisfacen un literal L_i, construido con el predicado *q*, puede calcularse como el producto de la densidad de *q* ([EC. 5.8]) multiplicada por N_i:

$$densidad(q) \cdot N_i$$

En el subconjunto de tuplas expandidas a partir de una concreta de T_1, el literal L_i será satisfecho, en media, por:

$$densidad(q) \cdot \prod_{j=1}^{i-1} crec\_med(L_j)$$

de estas tuplas. Por tanto, si se miden los parámetros de evaluación en el conjunto proyectado, bastará con que L_i sea satisfecho por una de estas tuplas para que no sea necesario evaluar con las restantes tuplas expandidas de la misma tupla original.

El número medio de tuplas expandidas a partir de una misma tupla de T_1, que deben usarse para evaluar un literal L_i (construido con el predicado q) será (puede verse la demostración en el apartado B.3):

$$\frac{1 + \prod_{j=1}^{i-1} crec\_med(L_j)}{1 + densidad(q) \cdot \prod_{j=1}^{i-1} crec\_med(L_j)} \qquad \text{[EC. 5.18]}$$

Partiendo de un conjunto T_1 con N_1 tuplas, la evaluación de cada literal L_i (con el predicado q) se realizará, en media, con un número de tuplas indicado por la siguiente expresión:

$$CE(L_i) = N_1 \cdot \frac{1 + \prod_{j=1}^{i-1} crec\_med(L_j)}{1 + densidad(q) \cdot \prod_{j=1}^{i-1} crec\_med(L_j)} \qquad \text{[EC. 5.19]}$$

Puede obtenerse una estimación del valor de CE medio a partir de la densidad media ponderada por el número de literales candidatos:

$$densidad\_med = \frac{\sum_{i=1}^{NumPrOrd} NumLit(Grado(q^i), u) \cdot densidad(q^i)}{\sum_{i=1}^{NumPrOrd} NumLit(Grado(q^i), u)} \qquad \text{[EC. 5.20]}$$

donde

q^i representa un predicado de la BD

Grado(q^i)	el grado del predicado q^i
NumLit(m,u)	el número de literales que pueden construirse para un predicado de grado *m* con un máximo de *u* variables usadas (apartado B.2)
densidad(q^i)	la densidad del predicado q^i ([EC. 5.8])
NumPrOrd	el número de predicados no borrosos de la BD

Con lo que resulta:

$$CE \approx N_1 \cdot \frac{1 + \prod_{j=1}^{i-1} crec\_med(L_j)}{1 + densidad\_med \cdot \prod_{j=1}^{i-1} crec\_med(L_j)} \qquad \text{[EC. 5.21]}$$

para la evaluación de cada literal candidato durante la búsqueda del antecedente i-ésimo de una cláusula en construcción, usando el heurístico *Interés** con el conjunto proyectado $T_1^{[i]}$.

El CE de FZFOIL para literales ordinarios sobre conjunto de tuplas no borrosas nunca puede ser mayor que el calculado para FOIL ([EC. 5.14]), ya que el caso peor es la evaluación de todas las tuplas (como ocurre en FOIL).

Para los literales borrosos (o sobre conjuntos de tuplas borrosas) es necesario evaluar con todas las tuplas, por lo que el coste de evaluación es el mismo que se calculó para FOIL ([EC. 5.14]). Esto es así porque el grado se satisfacción del literal candidato para una tupla proyectada (del conjunto $T_1^{[i]}$) vendrá dado por el máximo grado de satisfacción con todas las correspondientes tuplas expandidas (del conjunto T_i).

5.4.2.3 Conclusiones

El coste teórico (asociado al tamaño del espacio de búsqueda de literales) de FZFOIL es algo superior al obtenido en FOIL. El crecimiento de este CT se produce del siguiente modo:

- Linealmente respecto del número de predicados de la BD.
- Polinómicamente con el número de variables usadas en la cláusula.
- Exponencialmente con el grado de los predicados de la BD.

La diferencia de CT entre FOIL y FZFOIL consiste en que, en FZFOIL, el factor de crecimiento lineal asociado al número de predicados ordinarios es menor que el asociado al número de predicados borrosos.

El máximo coste de evaluación posible para FZFOIL es el mismo que se calculó para FOIL, por tanto viene limitado por la potencia de los literales que introducen variables nuevas. En el caso particular de que el conjunto local de tuplas sea no borroso y se evalúen literales ordinarios, el coste de evaluación decrece con la densidad de los predicados de la BD.

Capítulo 6:

CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

6.1 Resumen y conclusiones

Este trabajo presenta un sistema para aprendizaje de definiciones lógicas con incertidumbre, a partir de una base de datos relacional borrosa.

El campo de interés se centra, por tanto, en la programación lógica inductiva, introduciendo algunas interesantes aportaciones, principalmente en lo que se refiere a la entrada de datos y a los resultados producidos:

- Los datos de entrada pertenecen a una base de datos relacional borrosa. Por tanto, vienen expresados en forma de tablas de tuplas (relaciones), en las que las tuplas **pueden** llevar asociado un grado de pertenencia a la relación correspondiente. Se trata, por tanto, de relaciones borrosas, directamente identificables con conceptos borrosos (tan comunes en la realidad vista desde un punto de vista humano), y no de relaciones ordinarias con atributos borrosos (tal y como se entiende la "borrosidad" en muchos sistemas existentes).
- Los datos de salida vienen expresados en forma de definiciones lógicas de una relación (ordinaria o borrosa), que consta de una cláusula de Horn o de la disyunción de varias. Estas cláusulas de Horn se construyen mediante literales, aplicados sobre variables (generalmente), y asociados a relaciones borrosas u ordinarias. Los literales borrosos pueden ser modificados, además, por el empleo de etiquetas lingüísticas. Por tanto, se combina, en estas definiciones, la lógica de predicados con la lógica borrosa, en lo que podemos denominar "lógica borrosa de predicados", lo que constituye una aportación dentro de la inducción automática de conocimiento. Además, las definiciones inducidas llevan asociado un factor de incertidumbre, como hacen otros sistemas ya existentes.

El punto de partida del trabajo lo constituye un sistema de inducción de definiciones lógicas bien conocido: FOIL, creado por Quinlan en 1992, basado en la lógica de predicados. Sobre este sistema inicial se realizan, además de las extensiones para lógica borrosa ya mencionadas, otra serie de modificaciones y ampliaciones enfocadas a mejorar la inducción de conocimiento. Estas mejoras se realizan, principalmente, en su parte heurística, al definir una función de evaluación

de literales, basada en medidas de interés, que permite corregir algunas deficiencias del sistema original y aumentar la calidad de las reglas inducidas. Otras modificaciones se orientan hacia la introducción de conocimiento de base, mediante relaciones definidas intensionalmente, de modo similar a otros sistemas como FOCL.

Como resultado tangible de la tesis, se ha desarrollado y probado el sistema FZFOIL, disponible públicamente bajo la licencia GNU.

Con el sistema FZFOIL se consiguen mejorar las definiciones inducidas en algunos ejemplos con relaciones ordinarias, gracias a las modificaciones descritas en el capítulo 3 (nuevas funciones de evaluación y conjuntos locales de tuplas proyectadas). Estas mejoras pueden considerarse como la eliminación de algunos máximos locales durante la exploración del espacio de búsqueda. Sin embargo, no se garantiza la eliminación de otros máximos locales, que pueden provocar nuevos errores en la inducción de conocimiento.

La extensión de la lógica de primer orden hacia una lógica borrosa de primer orden (capítulo 4) permite, en FZFOIL, ampliar el campo de aplicación, al poder trabajar sobre BD relacionales que incluyan relaciones ordinarias y/o borrosas. La posibilidad de utilizar literales borrosos extiende el espacio de búsqueda de literales, ya que se permite aplicar etiquetas lingüísticas sobre los mismos, aumentando así la potencia descriptiva de las definiciones.

Dentro del proyecto SEIC se han realizado diferentes pruebas de FZFOIL, todas sobre datos reales, para describir el comportamiento de los usuarios de un sistema de información ciudadana. Los resultados de estas pruebas (capítulo 5) han servido para comparar las definiciones obtenidas sin utilizar lógica borrosa y definiciones borrosas inducidas a partir de relaciones borrosas. La conclusión extraída sobre este punto es que la inducción de definiciones borrosas da lugar, generalmente, a mejores resultados.

El análisis de la complejidad de FZFOIL demuestra que, aunque el espacio de búsqueda de literales crece algo más al existir relaciones borrosas (debido a las etiquetas lingüísticas de los literales borrosos), el tamaño del conjunto de entrenamiento no varía. Por el contrario, la complejidad de FZFOIL al ser aplicado sobre BD sin relaciones borrosas se simplifica, debido a la reducción del número de tuplas que es necesario evaluar (al trabajar con conjuntos de tuplas proyectadas).

6.2 Futuras líneas de investigación

6.2.1 Mejoras en algoritmos y heurísticos de FOIL y FZFOIL

6.2.1.1 Modificaciones en la función de evaluación

Puede ser interesante estudiar la incorporación de nuevas condiciones que debe cumplir la función de evaluación de literales *Interés*. Por ejemplo, que el interés de una regla decrezca al aumentar su complejidad ([EC. 3.4]), para favorecer la construcción de definiciones más cortas.

Otras posibilidades, algunas ya mencionadas en el apartado 3.3.1, son: costes especiales asignados al uso de ciertos predicados, coste por número de variables utilizadas en una cláusula, etc.

Merece la pena destacar otra alternativa interesante: aplicar la función de evaluación *Interés* no a literales candidatos sino a cláusulas parcialmente construidas. De este modo, se abren nuevas posibilidades para la exploración del espacio de búsqueda, como se propone en el siguiente apartado (6.2.1.2).

Es inmediato adaptar la función *RI* para calcular el interés de una cláusula de Horn C^{i-1}, mediante la siguiente asignación de A y B:

$A \equiv L_1 \wedge ... \wedge L_i$ (antecedentes de C^i)

$B \equiv P$ (literal objetivo de C^i)

resultando los siguientes argumentos, medidos en T_1:

N	$= N_1^+ + N_1^-$	número total de tuplas de T_1
N_A	$= N_1^{[i+1]+} + N_1^{[i+1]-}$	número de tuplas (+ y −) de T_1 que satisfacen $L_1 \wedge ... \wedge L_i$
N_B	$= N_1^+$	número de tuplas de T_1 que satisfacen el literal objetivo P
$N_{A \wedge B}$	$= N_1^{[i+1]+}$	número de tuplas de N_1 que satisfacen P y $L_1 \wedge ... \wedge L_i$ (se reduce a las tuplas + de $T_1^{[i+1]}$)

La expresión [EC. 3.3] aplicada sobre estos argumentos queda del siguiente modo:

$$\text{Interés}(C^i) = \frac{N_1^- \cdot N_1^{[i+1]+} - N_1^+ \cdot N_1^{[i+1]-}}{\sqrt{N_1^- \cdot N_1^+ \cdot (N_1^{[i+1]+} + N_1^{[i+1]-}) \cdot ((N_1^+ + N_1^-) - (N_1^{[i+1]+} + N_1^{[i+1]-}))}} \qquad \text{[EC. 6.1]}$$

siendo

$N_1^+ = |T_1^+|$

$N_1^- = |T_1^-|$

$N_1^{[i+1]+} = |T_1^{[i+1]+}|$

$N_1^{[i+1]-} = |T_1^{[i+1]-}|$

de este modo pueden evaluarse cláusulas candidatas en vez de literales candidatos. Esto abre la posibilidad de construir cláusulas por otros métodos, diferentes de la sucesiva evaluación y anexión del mejor literal candidato.

6.2.1.2 Nuevos algoritmos de búsqueda de descripciones

El algoritmo de búsqueda de descripciones utilizado en FOIL corresponde a un algoritmo del tipo "primero el mejor" (también denominado de "corta vista"), ya que, en cada iteración, selecciona el mejor literal candidato y lo añade a la cláusula en construcción. Además, establece algunos puntos de retroceso, en los que se puede bifurcar la búsqueda en caso de que por la rama elegida no se llegue a una definición válida (apartado 2.5.4.5).

Este algoritmo presenta, como ya se ha mencionado, algunas limitaciones importantes:

- Máximos locales.

  Este problema se debe a que, durante la evaluación de literales candidatos se selecciona siempre el mejor de ellos, descartando otros que, aun siendo peores inicialmente, pueden dar lugar a mejores definiciones en iteraciones posteriores. Los puntos de retroceso establecidos en FOIL no son siempre suficientes para garantizar la mejor definición y pueden presentarse

problemas como los analizados en el capítulo 3. En este capítulo también se proponía una nueva función de evaluación de literales (*Interés**), en sustitución de la original (*Ganancia*); de este modo se solventaban diferentes errores, posibles causantes de máximos locales, aunque no se garantizaba la eliminación de los mismos.

- Complejidad de la búsqueda.

  Como se deduce del análisis de la complejidad de FOIL (apartado 5.4), la construcción de nuevas cláusulas puede tener asociada una complejidad muy grande, especialmente en cláusulas con muchas variables. Muchas veces los heurísticos de poda aplicados (apartado 2.5.4.3) no son lo bastante eficientes como para limitar ésta suficientemente, y el coste de la búsqueda crece excesivamente.

Se han realizado algunos esfuerzos a solucionar el problema de los máximos locales mediante nuevos algoritmos de búsqueda ("algoritmo de los estados provisionales" de [Serrano, 95]). Estos algoritmos se basan en la exploración del espacio de búsqueda con uno o dos niveles de profundidad a partir de algunos literales. En algunos ejemplos se conseguía mejorar los resultados obtenidos, aunque en otros no sólo no se mejoraba, sino que el tiempo de ejecución crecía considerablemente. En cualquier caso, este algoritmo tampoco garantiza la eliminación de máximos locales.

Para simplificar la complejidad de la búsqueda, algunos sistemas, como INDUCE ([Dietterich y Michalski, 84]) o RDT/DB ([Morik y Brockhausen, 97]), realizan una búsqueda en dos pasos, mediante la partición del espacio de descripciones en dos niveles (denominados de *sólo estructura* y de *atributos* en INDUCE). Adaptando esta idea a sistemas de ILP, podría simplificarse la complejidad de la construcción de definiciones.

Otro enfoque que se plantea (que puede coexistir con la "búsqueda en dos niveles") es la posibilidad de usar algoritmos de búsqueda paralela, que mejoren los resultados ante el problema de los máximos locales y cuya complejidad crezca de forma controlada. Siguiendo esta línea, puede pensarse en la utilización de algoritmos genéticos para explorar el espacio de descripciones.

La utilización de algoritmos genéticos representa una estrategia de búsqueda alternativa, que permite evitar algunas de las principales deficiencias de la búsqueda de "corta vista" o "hill-climbing" usada en FOIL, como la caída en máximos locales. Al mismo tiempo, resulta más eficiente que la búsqueda exhaustiva. El inconveniente de estos métodos es que no garantizan que se encuentre la solución óptima, aunque suelen dar buenos resultados.

En la literatura pueden encontrarse algunos sistemas que utilizan algoritmos genéticos para construir definiciones:

- GABIL ([De Jong et al., 93]), orientado a la lógica de proposiciones, codifica de una forma uniforme el valor de los atributos y los antecedentes de una cláusula.
- REGAL ([Neri y Saitta, 96]), utiliza, al igual que el anterior, una codificación uniforme, mediante cadenas de bits de longitud fija, para las diferentes cláusulas en forma DNF.
- GLPS ([Wong y Leung, 95]). Este sistema permite inducir programas lógicos expresados en lógica de primer orden. Los programas lógicos que construye los representa en forma de árbol, con nodos AND y OR, en el que las hojas son predicados aplicados sobre términos variables.

Para abordar la búsqueda de descripciones con un algoritmo genético se propone construir una población de cláusulas de Horn candidatas, todas con la misma cabeza, en la que se apliquen operadores genéticos de selección (aleatoria o ponderada por el valor de evaluación de los individuos), entrecruzamiento (intercambio de antecedentes entre pares de cláusulas) y mutación (modificación de algún antecedente de una cláusula). De este modo, se generan nuevos individuos en la población (nuevas cláusulas), algunos de los cuales mejorarán las descripciones obtenidas.

Para poder evaluar las diferentes cláusulas candidatas, se puede adaptar fácilmente la función de evaluación *Interés* para su utilización con cláusulas ([EC. 6.1]).

6.2.2 Ampliaciones en el lenguaje de representación del conocimiento

La lógica de predicados ofrece una gran potencia descriptiva, aunque generalmente en los sistemas de programación lógica inductiva, en particular en FOIL, se emplea una versión restringida de la misma, ya que no se permite la utilización de funciones dentro de las definiciones lógicas.

En FZFOIL se extiende la potencia descriptiva de la lógica de predicados hacia la lógica borrosa de predicados, permitiendo describir relaciones borrosas (y ordinarias) a partir de otras relaciones (borrosas u ordinarias), y calificar los literales borrosos con etiquetas lingüísticas. Sin embargo, las cláusulas construidas tampoco permiten la utilización de funciones.

Por otro lado, FOIL es incapaz de aumentar su vocabulario mediante la invención de nuevos predicados. En FZFOIL ocurre lo mismo, a pesar de que la utilización de etiquetas lingüísticas permite modificar, en cierta medida, el significado de los predicados borrosos.

Podría resultar de interés extender la potencia descriptiva de FZFOIL en ambas direcciones: por un lado, incorporar la utilización de funciones en las definiciones, y por otro, considerar la posibilidad de inventar nuevos predicados. Existen sistemas de ILP, como SIERES ([Wirth y O'Rorke, 91]) o CHILLIN ([Zelle et al., 94]), que incorporan ambas facilidades, aunque, por otro lado sufren de otras importantes limitaciones (manejo de datos ruidosos, definiciones recursivas, atributos sin tipo asignado, etc.).

6.2.3 Mayor flexibilidad en la representación de conocimiento borroso

En el sistema FZFOIL desarrollado se permite introducir relaciones borrosas, en las que sus tuplas tienen asignado un grado de pertenencia. A partir de estas relaciones se construyen literales borrosos, eventualmente con etiquetas lingüísticas, para construir las definiciones lógicas. Este esquema se ajusta a algunos modelos propuestos para bases de datos relacionales borrosas (modelo de Umano, apartado 2.6.4.2), aunque algunos otros (modelo de Buckles y Petry, o modelo de Medina) añaden la borrosidad a los atributos, considerando los dominios como conjuntos borrosos.

En FZFOIL se permite utilizar atributos borrosos en las relaciones de la base de datos, aunque no se consideran más que como valores de un dominio de tipo nominal, ya que, a través del grado de pertenencia de la correspondiente tupla, no es posible distinguir unos de otros. Lo mismo ocurre cuando estos atributos se utilizan como constantes dentro de un literal borroso, en una definición.

Resultaría interesante poder manejar estos valores borrosos como tales, desde el punto de vista semántico, de modo que, en las definiciones construidas, se pudiesen utilizar relaciones de equivalencia establecidas para sus dominios. En algunos modelos de base de datos relacional

borrosa (Buckles y Petry) se permiten definir estas clases de equivalencia, que indican la similitud entre diferentes etiquetas de un dominio borroso.

En el modelo de Medina ([Medina et al., 94]) lo que se propone es la utilización de diferentes formas de representar la incertidumbre en los atributos de una relación: mediante distribuciones de posibilidad, etiquetas lingüísticas, funciones, etc. Extender la funcionalidad de FZFOIL en este sentido, para manejar "relaciones borrosas generalizadas", como las del modelo mencionado, aumentaría la complejidad del algoritmo (el espacio de búsqueda de literales crecería considerablemente), aunque también aumentaría la potencia descriptiva de las definiciones borrosas. En cualquier caso, sería interesante disponer de resultados de este tipo para evaluarlos de forma precisa.

6.2.4 Mejoras en el proceso de inducción

Aunque sistemas como FOIL, FZFOIL, etc. ya han demostrado su validez en algunos ejemplos reales, es indudable que aún requieren un gran esfuerzo del ingeniero de conocimiento. El éxito de estos sistemas está, muchas veces, en la forma en que se seleccionan, preprocesan y filtran los datos de partida. Pero quizá la limitación más importante está en que se trata de sistemas de aprendizaje inductivo con ejemplos (apartado 2.3.2.2), en los que es necesario especificar la relación o relaciones objetivo, para las que se quieren construir definiciones lógicas.

Resulta mucho más prometedora la sugerencia de convertir estos sistemas en otros capaces de realizar *descubrimiento* en bases de datos. Un sistema de descubrimiento resulta más interesante, al permitir obtener definiciones generales a partir de las relaciones almacenadas en la base de datos, sin conocer a priori cuál es la relación objetivo. Una sugerencia más ambiciosa es que, además, se puedan *formar conceptos*, entendiendo como tales subconjuntos interesantes de alguna relación, con definiciones lógicas sencillas construidas a partir de otras relaciones. Siguiendo esta línea hay que mencionar algunos sistemas como CHILLIN ([Zelle et al., 94]), CIGOL ([Muggleton y Buntime, 88]), etc., capaces de inventar nuevos predicados.

Apéndice A:

CONCEPTOS DE LÓGICA BORROSA

El término “lógica borrosa” (*fuzzy logic*) puede interpretarse como ([Kantrowitz et al., 94]) un superconjunto de la tradicional lógica booleana, que ha sido extendida para manejar el concepto de “parcialmente verdadero” (valores de verdad entre “absolutamente verdadero” y “absolutamente falso”)[1]. Fue presentada por Lotfi Zadeh de UC/Berkeley en los años 60 ([Zadeh, 65], [Zadeh, 73]), como un medio para modelar la incertidumbre del lenguaje natural.

Según Zadeh , no debería considerarse la teoría borrosa como una simple teoría, sino que se debería considerar el proceso de *borrosificación* (en inglés *fuzzification*) como una metodología para generalizar cualquier teoría desde su versión ordinaria (discreta) a una nueva versión continua (borrosa) (véase el *principio de extensión*, apartado A.1.3.7). Así puede hablarse de “cálculo borroso”, “ecuaciones diferenciales borrosas”, “autómatas borrosos” ([Klir y Yuan, 95]), “sistemas dinámicos borrosos”, etc.

Del mismo modo que se verifica una estrecha relación entre la lógica booleana y el concepto de subconjunto, así también se cumple una relación similar entre la lógica borrosa y la teoría de conjuntos borrosos (apartado A.1).

A.1 Conjuntos borrosos y conceptos relacionados

A.1.1 Función de pertenencia y conjunto de pertenencia para un subconjunto borroso

En la teoría clásica de conjuntos, dado un elemento *x* de un universo *U* y un subconjunto ordinario $A \subseteq U$, puede definirse una *función característica* χ_A que define qué elementos de U pertenecen al conjunto A y cuáles no ([Klir y Yuan, 95]), del siguiente modo:

1. Una frecuente mala interpretación de la lógica borrosa se debe a que ésta puede considerarse, en un sentido muy restringido, como una extensión y generalización de las clásicas lógicas multivaloradas para infinitos valores de verdad. Sin embargo, la lógica borrosa va más allá, porque no sólo considera que hay una infinidad de valores semánticos entre “verdadero” y “falso”, sino que también tiene en cuenta que esos mismos valores de verdad son imprecisos ([Fernández y Sáez-Vacas, 87]).

$$\chi_A(x) = \begin{cases} 1, \text{ para } x \in A \\ 0, \text{ para } x \notin A \end{cases} \qquad \text{[EC. A.1]}$$

de este modo, la función característica de A asigna a cada elemento de U, un elemento del conjunto {0, 1}:

$$\chi_A : U \rightarrow \{0, 1\} \qquad \text{[EC. A.2]}$$

La función característica cumple las siguientes propiedades para la complementación, intersección y unión de conjuntos:

$$\chi_{\overline{A}}(x) = 1 - \chi_A(x)$$

$$\chi_{A \cap B}(x) = \chi_A(x) \cdot \chi_B(x)$$

$$\chi_{A \cup B}(x) = \chi_A(x) + \chi_B(x) \text{(suma lógica)}$$

Para un universo *U* se define un *subconjunto borroso* $\underline{A}$ de *U* como un conjunto de pares de la forma:

$$\underline{A} = \{ <x, \mu_A(x)> \}, \forall x \in U \qquad \text{[EC. A.3]}$$

donde $\mu_A(x)$ es una *función de pertenencia* que toma sus valores en un conjunto *M*, llamado *conjunto de pertenencia*, que generalmente es el intervalo cerrado de reales entre 0 y 1:

$$M = [0, 1] \qquad \text{[EC. A.4]}$$

Por tanto, la función de pertenencia de un conjunto borroso $\underline{A}$ no es más que una extensión de la función característica definida para conjuntos ordinarios ([EC. A.2]), ya que si se hace M = {0, 1}, entonces $\underline{A}$ se reduce a un subconjunto ordinario (la teoría clásica de conjuntos es un caso particular de la teoría de conjuntos borrosos):

$$\mu_A : U \rightarrow [0, 1] \qquad \text{[EC. A.5]}$$

En la literatura se suelen emplear dos notaciones distintas para la función de pertenencia de un conjunto borroso:

- La primera de ellas, más tradicional, utiliza el símbolo μ_A para representar la función de pertenencia al subconjunto borroso $\underline{A}$, tal como hemos visto.
- La otra notación utilizada expresa la función de pertenencia como $\underline{A}$, es decir:

  $$\underline{A} : U \rightarrow [0, 1] \qquad \text{[EC. A.6]}$$

  de modo que se usa un mismo símbolo tanto para denominar al conjunto borroso como para su función de pertenencia; según indican ([Klir y Yuan, 95]), no existe ambigüedad en este doble uso del mismo símbolo: cada conjunto borroso queda completa y unívocamente definido por una función de pertenencia, por lo que los símbolos de las funciones de pertenencia podrían también usarse en los conjuntos borrosos asociados.

Dado que una relación biunívoca no es necesariamente una identidad (en sentido matemático), no exime de ambigüedad, por lo que utilizaremos la primera notación: $\underline{A}$ para el subconjunto borroso y μ_A para su función de pertenencia.

A.1.2 Igualdad e inclusión de conjuntos borrosos

Se definen las relaciones de igualdad e inclusión entre dos subconjuntos borrosos $\underline{A}$ y $\underline{B}$ del siguiente modo:

Igualdad:

$$\underline{A} = \underline{B} \quad \Leftrightarrow \quad \mu_A(x) = \mu_B(x),\ \forall x \in U \qquad \text{[EC. A.7]}$$

Inclusión:

$$\underline{A} \subseteq \underline{B} \quad \Leftrightarrow \quad \mu_A(x) \leq \mu_B(x),\ \forall x \in U \qquad \text{[EC. A.8]}$$

A.1.3 Operaciones entre conjuntos borrosos

A.1.3.1 Complementación

Se define del siguiente modo:

$$\underline{A} = \overline{\underline{B}} \Leftrightarrow \mu_A(x) = 1 - \mu_B(x),\ \forall x \in U \qquad \text{[EC. A.9]}$$

suponiendo que M = [0, 1].

A.1.3.2 Intersección

La intersección de conjuntos borrosos se define del siguiente modo[1]:

$$\underline{C} = \underline{A} \cap \underline{B} \Leftrightarrow \mu_C(x) = \min(\mu_A(x)\ ,\ \mu_B(x)),\ \forall x \in U \qquad \text{[EC. A.10]}$$

A.1.3.3 Unión

La unión de conjuntos borrosos se define así:

---

1. La unión e intersección de conjuntos borrosos se definen, en general, como funciones de la forma:

$f: [0, 1] \times [0, 1] \rightarrow [0, 1]$

t.q. $\mu_{f\,[A,B]}(x) = f\,[\mu_A(x)\ ,\ \mu_B(x)]$

que debe cumplir ciertos axiomas (véase [Klir y Folger, 88]). Existen diferentes clases de funciones cuyos elementos satisfacen todos los requerimientos impuestos a estas funciones borrosas (funciones de: Yager, Schweizer & Sklar, Hamacher, Frank, Dombi, etc.).

Por ejemplo, la clase de funciones de Yager para la unión de conjuntos borrosos viene definida por la expresión:

$u_w(a, b) = \min\,[1, (a^w + b^w)^{1/w}]$, donde $w \in (0, \infty)$

Dentro de esta clase, tiene especial interés la función obtenida para el valor $w = \infty$, pues es la clásica definición de la unión borrosa:

$u_\infty(a, b) = \lim_{(w \rightarrow \infty)} \{\ \min[1, (a^w + b^w)^{1/w}]\ \} = \max(a, b)$

De modo análogo, de la clase de funciones de intersección de Yager, para el valor $w = \infty$, se obtiene la clásica definición de la intersección borrosa:

$i_\infty(a, b) = \lim_{(w \rightarrow \infty)} \{\ \min[1, ((1-a)^w + (1-b)^w)^{1/w}]\ \} = \min(a, b)$

$$\underline{C} = \underline{A} \cup \underline{B} \Leftrightarrow \mu_C(x) = \max(\mu_A(x) , \mu_B(x)), \forall x \in U \quad \text{[EC. A.11]}$$

A.1.3.4 Producto

El producto de conjuntos borrosos se define así:

$$\underline{C} = \underline{A} \cdot \underline{B} \Leftrightarrow \mu_C(x) = \mu_A(x) \cdot \mu_B(x), \forall x \in U \quad \text{[EC. A.12]}$$

A.1.3.5 Potenciación

La potenciación de conjuntos borrosos se define así:

$$\underline{C} = \underline{A}^{\alpha} \Leftrightarrow \mu_C(x) = \mu_A{}^{\alpha}(x), \forall x \in U \quad \text{[EC. A.13]}$$

A.1.3.6 Producto cartesiano

Se define el producto cartesiano entre conjuntos borrosos del siguiente modo:

$$\underline{A} \times \underline{B} = \{< (a, b), \min(\mu_A(a), \mu_B(b))>\} \quad \text{[EC. A.14]}$$

siendo

$a \in \underline{A} \subseteq U_A$

$b \in \underline{B} \subseteq U_B$

Por lo tanto, es un subconjunto borroso del producto cartesiano de los universos U_A y U_B.

A.1.3.7 Principio de extensión para conjuntos borrosos

El *principio de extensión* para conjuntos borrosos ([Klir y Yuan, 95]) establece que, para cualquier función ordinaria $f : X \rightarrow Y$, pueden inducirse dos funciones de la siguiente forma:

$f : F(X) \rightarrow F(Y)$

y su inversa f^{-1}:

$f^{-1} : F(Y) \rightarrow F(X)$

que se definen como:

$$\mu_{f(A)}(y) = \max \{ \mu_A(x) \mid f(x) = y\}, \quad \forall A \in F(X)$$
$$\mu_{f^{-1}(B)}(x) = \mu_B(f(x)), \quad \forall B \in F(Y)$$

donde F (*X*) y F (*Y*) son los conjuntos de todos los subconjuntos borrosos (*fuzzy power sets*) de *X* e *Y* respectivamente.

Se suele usar el mismo símbolo *f*, para la función ordinaria (en los universos *X* e *Y*) y para su extensión borrosa (en los universos F (*X*) y F (*Y*)).

Según este principio, tanto las relaciones de igualdad e inclusión como las operaciones de complementación, intersección y unión definidas para conjuntos

borrosos se reducen a las definiciones clásicas en conjuntos ordinarios (para M = {0, 1}). Del mismo modo, todas las propiedades de estas operaciones en los conjuntos ordinarios (asociatividad, distributividad, etc.) se siguen cumpliendo en los conjuntos borrosos, excepto dos de ellas:

$$\underline{A} \cap \overline{\underline{A}} \neq \varnothing \qquad \text{[EC. A.15]}$$

$$\underline{A} \cup \overline{\underline{A}} \neq U \qquad \text{[EC. A.16]}$$

A.1.4 Algunas definiciones útiles

A.1.4.1 Soporte

Se define el soporte (*support*) de un conjunto borroso $\underline{A}$ en el conjunto universal *U*, como el conjunto ordinario formado por todos los elementos de *U* cuyo grado de pertenencia en $\underline{A}$ es distinto de cero:

$$\text{supp}(\underline{A}) = \{ x \in U \mid \mu_A(x) > 0 \} \qquad \text{[EC. A.17]}$$

A.1.4.2 Altura

Se define la altura (*height*) de un conjunto borroso $\underline{A}$ como el mayor grado de pertenencia de todos los elementos en dicho conjunto:

$$h(\underline{A}) = \max \{ \mu_A(x) \}, \forall x \in U \qquad \text{[EC. A.18]}$$

La altura de los conjuntos normalizados será 1.

A.1.4.3 Corte - alpha

Un corte alpha (α-*cut*) de un conjunto borroso $\underline{A}$ es un conjunto ordinario ${}^{\alpha}A$ formado por todos los elementos del conjunto universal *U* cuyos grados de pertenencia en $\underline{A}$ son mayores o iguales que el valor α:

$${}^{\alpha}A = \{ x \in U \mid \mu_A(x) \geq \alpha \} \qquad \text{[EC. A.19]}$$

A.1.4.4 Conjunto de niveles

Se denomina conjunto de niveles (*level set*) de un conjunto borroso $\underline{A}$, y se representa como $\Lambda(A)$, al conjunto de grados de pertenencia de sus elementos:

$$\Lambda(\underline{A}) = \{ \alpha \mid \mu_A(x) = \alpha \text{ para algún } x \in U \} \qquad \text{[EC. A.20]}$$

A.1.4.5 Cardinalidad escalar

La cardinalidad escalar (*scalar cardinality*) de un conjunto borroso $\underline{A}$ definido en un conjunto universal *U* finito es la suma de los grados de pertenencia de todos los elementos de *U* en $\underline{A}$:

$$|\underline{A}| = \sum_{x \in U} \mu_A(x) \qquad \text{[EC. A.21]}$$

A.2 Determinación de los grados de pertenencia

La utilidad de un conjunto borroso para modelar un concepto o una etiqueta lingüística dependerá de la forma que tome su función de pertenencia. Por ello, resulta de gran importancia la determinación práctica de una función de pertenencia precisa y justificable. Los métodos más usados son empíricos y se basan en experimentos realizados en una población, para medir la percepción subjetiva de los grados de pertenencia a la clase conceptual que se quiere modelar. Existen varios métodos:

1. Evaluación subjetiva: un individuo asigna un grado de pertenencia subjetivo a cada elemento; normalmente esta evaluación la realizan expertos en el tema o aplicación de que se trate.
2. Métodos psicológicos: para aplicaciones complejas se suelen aplicar métodos psicológicos, como medir el tiempo de respuesta requerido para clasificar un elemento (respuestas más rápidas se considera que indican grados de pertenencia mayores), etc.
3. Frecuencias o probabilidades: estadísticas basadas en histogramas o el porcentaje de respuestas afirmativas y negativas sobre la pertenencia de un elemento al conjunto[1].
4. Funciones *ad-hoc*: en los sistemas borrosos de control se suele utilizar un pequeño conjunto de sencillas funciones (por ejemplo funciones triangulares o en forma de trapecio) como funciones de pertenencia. De este modo, el problema se reduce a la elección de unos pocos parámetros en dichas funciones.

A los conjuntos borrosos en los que la función de pertenencia asigna a sus elementos valores de pertenencia que son números reales, se les denomina *conjuntos borrosos de tipo 1*. Es posible extender el concepto de conjunto borroso a los grados de pertenencia para construir así *conjuntos borrosos de tipo 2*, en los que los grados de pertenencia de sus elementos serían, a su vez, conjuntos borrosos. Aplicando de forma recursiva esta idea, se podrían construir, de forma general, conjuntos borrosos de *tipo L*. Por ejemplo, si definimos el concepto “inteligente” como un conjunto borroso de tipo 1, asignaríamos a cada persona un grado de pertenencia que sería un número real (por ejemplo, su cociente intelectual normalizado). Pero si definimos el concepto “inteligente” como un conjunto borroso de tipo 2, entonces el grado de pertenencia asignado a cada persona sería a su vez un conjunto borroso (por ejemplo, las etiquetas “superdotado”, “por encima de la media”, “normal”, “por debajo de la media”, “idiota”, etc.).

A.3 Relaciones ordinarias vs. relaciones borrosas

Una relación representa una conexión o correspondencia entre dos o más elementos. Una *relación ordinaria* representa la existencia o ausencia de una asociación, interacción o

1. Como ya se ha mencionado, los grados de pertenencia a un conjunto borroso no son (necesariamente) probabilidades. Sin embargo, cualquier distribución de probabilidad puede interpretarse como un conjunto borroso.

interconexión entre los elementos de dos o más conjuntos. Este concepto se puede generalizar para permitir diversos grados de fuerza en la relación entre elementos. Los diferentes grados de asociación pueden representarse mediante grados de pertenencia a una *relación borrosa*, del mismo modo que el grado de pertenencia de un elemento a un conjunto borroso.

Por analogía, así como un conjunto ordinario puede verse como un caso particular de un conjunto borroso, una relación ordinaria puede considerarse como un caso particular de relación borrosa.

Una *relación ordinaria* entre dos conjuntos *A* y *B* se define por el conjunto de pares ordenados *(a,b)* que la verifican:

$$R = \{ (a, b) \}, \quad a \in A, b \in B \qquad \text{[EC. A.22]}$$

La relación así definida es binaria, aunque el mismo concepto se puede generalizar para formar relaciones de orden *n*, definidas por conjuntos de tuplas de grado *n*.

Se puede observar que una relación no es más que un subconjunto del producto cartesiano de los conjuntos que intervienen:

$$R \subseteq A_1 \times A_2 \times ... \times A_n \qquad \text{[EC. A.23]}$$

donde $U = A_1 \times A_2 \times ... \times A_n$ representa el *conjunto universal* o *universo del discurso* para R. Debido a que una relación es realmente un conjunto, conceptos como inclusión, unión, intersección y complemento de conjuntos pueden ser aplicados directamente a relaciones.

Se dice que la relación R es *ordinaria* (no borrosa) cuando el conjunto de tuplas que la define es un subconjunto ordinario (no borroso) del producto cartesiano, con independencia de si los conjuntos A_i son ordinarios o borrosos.

Por ejemplo, para las relaciones de igualdad e inclusión entre conjuntos borrosos, definidas en el apartado A.1.2, dados dos subconjuntos borrosos, $\underline{A}$ y $\underline{B}$, se cumple que:

$$\underline{A} = \underline{B}, \text{ o bien que } \underline{A} \neq \underline{B}$$

y que

$$\underline{A} \subset \underline{B}, \text{ o bien que } \underline{A} \not\subset \underline{B}$$

Por tanto, a pesar de que estas relaciones están definidas sobre conjuntos borrosos, no son borrosas.

De modo similar, podemos definir una *relación borrosa* $\underline{R}$ como un subconjunto (borroso) de las tuplas de U, es decir, en el que cada tupla tiene asociado un grado de pertenencia a $\underline{R}$:

$$\underline{R} \subseteq A_1 \times A_2 \times ... \times A_n \qquad \text{[EC. A.24]}$$

correspondiendo en este caso el signo "$\underline{\subseteq}$" a la operación de inclusión entre conjuntos borrosos (apartado A.1.2).

Se define una función de pertenencia, que expresa el *grado de pertenencia* de cada tupla t a la relación $\underline{R}$, o dicho de otro modo, la fuerza de la relación $\underline{R}$ existente entre los atributos de t. El grado de pertenencia a una relación borrosa puede tomar valores en el intervalo [0, 1]:

$$\mu_{\underline{R}}: U \rightarrow [0, 1]$$

La relación borrosa $\underline{R}$ se puede definir extensionalmente como el conjunto de pares de la forma:

$$\underline{R} = \{<t, \mu_{\underline{R}}(t)> \mid t \in U, \mu_{\underline{R}}(t) \in [0, 1]\} \qquad \text{[EC. A.25]}$$

siendo t una tupla del conjunto universal U, y $\mu_{\underline{R}}(t)$ el grado de pertenencia[1] de t a $\underline{R}$.

A.3.1 Relaciones borrosas entre conjuntos ordinarios

Una *relación borrosa* $\underline{R}$ entre dos conjuntos ordinarios *A* y *B* se define como un conjunto de pares ordenados *(a, b)*, cada uno con un determinado grado de pertenencia μ_R a la relación $\underline{R}$:

$$\underline{R} = \{<(a, b), \mu_R(a, b)>\},\ a \in A,\ b \in B,\ \mu_R \in [0, 1] \qquad \text{[EC. A.26]}$$

donde μ_R indica en qué grado, o con qué intensidad, los elementos *a* y *b* están en la relación $\underline{R}$. Una forma de representar una relación binaria borrosa es mediante la matriz de pertenencia, como veremos en el siguiente ejemplo ([Fernández y Sáez-Vacas, 95]):

> La relación que existe entre las estaciones del año y la sensación de calor o frío que se siente es una relación borrosa (pues representa una relación subjetiva). Una persona podría expresarla explícitamente como:
>
> A = {primavera, verano, otoño, invierno}
>
> B = {calor, frío}

	calor	frío
primavera	0.7	0.4
verano	1	0
otoño	0.6	0.5
invierno	0.1	1

Del mismo modo que ocurre con las relaciones ordinarias, el concepto de relación borrosa se puede generalizar a órdenes superiores (entre más de dos conjuntos).

Se puede observar que la relación borrosa $\underline{R}$ no es más que un subconjunto borroso del producto cartesiano de *A* y *B*:

$$\underline{R} \subseteq A \times B$$

y, en general, una relación borrosa n-aria entre *n* conjuntos ordinarios: A_1, A_2,..., A_n será un subconjunto borroso del producto cartesiano de todos ellos:

$$\underline{R} \subseteq A_1 \times A_2 \times ... \times A_n \qquad \text{[EC. A.27]}$$

Al ser los conjuntos A_i ordinarios, su producto cartesiano consta de todas las tuplas $(a_1, a_2,..., a_n)$, y la relación borrosa $\underline{R}$ le asigna un grado de pertenencia a cada una de ellas.

1. Como se comenta en el apartado A.1.1, hay que destacar dos posibles notaciones para el grado de pertenencia a una relación borrosa (conjunto borroso en general): en una de ellas se utiliza el mismo símbolo $\underline{P}$ con que se representa a la relación borrosa; en la otra, por el contrario, se utiliza el símbolo μ_P para el grado de pertenencia a la relación $\underline{P}$. Aquí seguiremos esta segunda notación, que, por otro lado, es la más tradicional.

A.3.2 Relaciones borrosas entre conjuntos borrosos

Una relación borrosa entre dos conjuntos borrosos $\underline{A}$ y $\underline{B}$ se define como:

$$\underline{R} = \{ <(a, b), \mu_R(a, b)> \} \qquad \text{[EC. A.28]}$$

siendo

$$a \in \underline{A}, \quad b \in \underline{B}, \qquad \mu_R(a, b) \leq \min(\mu_A(a), \mu_B(b))$$

que no es más que un subconjunto de su producto cartesiano (apartado A.1.3.6). Cualquier subconjunto de $\underline{A} \times \underline{B}$ será una relación borrosa entre ambos. La generalización al caso n-ario es inmediata:

$$\underline{R} \subseteq \underline{A}_1 \times \underline{A}_2 \times ... \times \underline{A}_n \qquad \text{[EC. A.29]}$$

A.3.3 Composición de relaciones

Sean dos relaciones binarias R_1 y R_2, definidas entre los conjuntos A y B, y B y C respectivamente:

$$R_1 = \{<(a, b), \mu_{R1}(a, b)>\}, \quad a \in A, b \in B$$

$$R_2 = \{<(b, c), \mu_{R2}(b, c)>\}, \quad b \in B, c \in C$$

Se define la relación compuesta de R_1 y R_2 del siguiente modo:

$$R_1 \circ R_2 = \{<(a, c), \max_b[\min(\mu_{R1}(a,b), \mu_{R2}(b,c)))]\} \qquad \text{[EC. A.30]}$$

es decir, para cada valor de $b \in B$ se toma el mínimo de las funciones de pertenencia a las relaciones R_1 y R_2 de las parejas (a,b) y (b,c) respectivamente; el máximo de todos ellos es el grado de pertenencia a $R_1 \circ R_2$ de la pareja (a,c).

A.4 Proposiciones borrosas

La principal diferencia entre las proposiciones construidas en la clásica lógica booleana y las proposiciones borrosas está en los grados de verdad que se asignan a las mismas, dados por la función de evaluación E. En el caso booleano, una proposición p puede ser verdadera o falsa únicamente, representando estos valores como E(p) = 1 (verdadera) y E(p) = 0 (falsa). En el caso borroso, la función de evaluación puede tomar valores en el intervalo unidad:

$$E(p) \in [0, 1] \qquad \text{[EC. A.31]}$$

Veremos a continuación cuatro tipos de proposiciones borrosas ([Klir y Yuan, 95]):

- proposiciones no condicionales y no calificadas
- proposiciones no condicionales y calificadas
- proposiciones condicionales y no calificadas
- proposiciones condicionales y calificadas

A.4.1 Proposiciones borrosas no condicionales y no calificadas

La forma canónica de una proposición borrosa p de este tipo es la siguiente:

$$p: X \text{ es } \underline{B} \qquad \text{[EC. A.32]}$$

siendo *X* una variable que toma valores x en algún conjunto universal U, y $\underline{B}$ es un subconjunto borroso definido en U.

El grado de verdad asociado a la proposición p se interpreta como el grado de pertenencia al conjunto borroso $\underline{B}$. Así para un valor concreto x de la variable *X*, se tiene que

$$E(p) = \mu_B(x) \qquad \text{[EC. A.33]}$$

siendo E() la función de evaluación para las proposiciones.

En algunas proposiciones borrosas, los valores de la variable *X* son asignados a individuos de un conjunto I; en ese caso, la variable *X* se transforma en una función X: I $\rightarrow$ U, quedando la definición de la proposición borrosa ([EC. A.32]) como:

$$p: X(i) \text{ es } \underline{B}, \text{ siendo } i \in I \qquad \text{[EC. A.34]}$$

Por ejemplo, si I es el conjunto de personas, cada persona está caracterizada por su edad, y está definido el conjunto borroso *Joven*, podemos representar la proposición "i es joven" como:

p: Edad(i) es Joven

El grado de verdad de esta proposición vendrá dado por el grado de pertenencia al conjunto *Joven* para la edad de cada individuo:

$$E(p) = \mu_{Joven}(Edad(i))$$

Por ejemplo, en la figura A-2 se define, para una edad de 25 años, un grado de pertenencia de 0.87 al conjunto borroso *Joven*.

A.4.2 Proposiciones borrosas no condicionales y calificadas

Podemos distinguir dos tipos de proposiciones borrosas calificadas, dependiendo del tipo de calificador o modificador que se utilice:

- Proposiciones con *calificador borroso de verdad*, o *etiqueta lingüística*, que modifica el grado de verdad de la proposición sin calificar;
- Proposiciones con *calificador borroso de probabilidad*, que representan distribuciones de probabilidad para un conjunto borroso.

A.4.2.1 Proposición borrosa con calificador de verdad

La forma canónica de una proposición p de este tipo es la siguiente:

$$p: X \text{ es } \underline{B} \text{ es V} \qquad \text{[EC. A.35]}$$

donde *X* es una variable que toma valores en un conjunto universal U, $\underline{B}$ es un subconjunto borroso definido en U, y V es un calificador borroso de verdad.

Definimos un *calificador borroso de verdad* V como una función que modifica el grado de verdad de una proposición borrosa p. El grado de verdad E(p) de la proposición p con calificador

borroso de verdad se calcula como el resultado de aplicar una función V, que representa al calificador, sobre el grado de pertenencia al conjunto borroso B:

$$E(p) = V(\mu_B(x)), \qquad \forall x \in U \qquad \text{[EC. A.36]}$$

Los calificadores borrosos de verdad son etiquetas lingüísticas aplicadas a los valores de verdad, por ejemplo: “verdadero”, “muy verdadero”, “algo verdadero”, “muy falso”, etc. y las funciones asociadas pueden verse en la figura A-1.

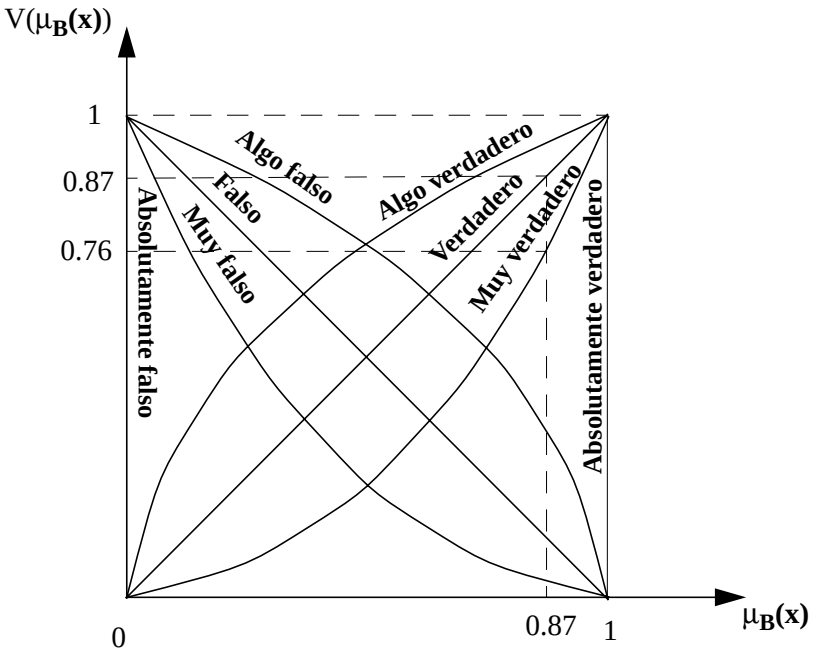

Figura A-1: Ejemplo de calificadores de grado de verdad borrosos

Los calificadores borrosos de verdad definen *valores de verdad lingüísticos*, que representan subconjuntos borrosos dentro del conjunto de pertenencia [0, 1]:

$V \in$ {verdadero, falso, no verdadero, no falso, bastante verdadero, bastante falso,
poco verdadero, poco falso, muy verdadero, más o menos verdadero, ...}

Por ejemplo, la proposición borrosa “Tina es joven” se puede considerar que equivale a “Tina es joven es verdadero”. Si suponemos que *Tina* pertenece al conjunto representado por el predicado *joven* con un grado de pertenencia de 0.87, entonces el grado de verdad de la proposición “Tina es joven” será también 0.87 (figura A-1).

Pero podríamos construir otras proposiciones como “Tina es joven es muy verdadero”, “Tina es joven es bastante verdadero”, etc. cuyos grados de verdad ya no coinciden con el grado de pertenencia de *Tina* al conjunto *joven*, ya que los cualificadores borrosos “muy verdadero”, “bastante verdadero”, etc. afectan al significado de la proposición inicial “Tina es joven”. Si modificaramos directamente los predicados “muy joven”, “bastante joven”, etc. obtendríamos proposiciones (“Tina es muy joven”, “Tina es bastante joven”, etc.) cuyos grados de verdad habrían cambiado del mismo modo (figura A-2).

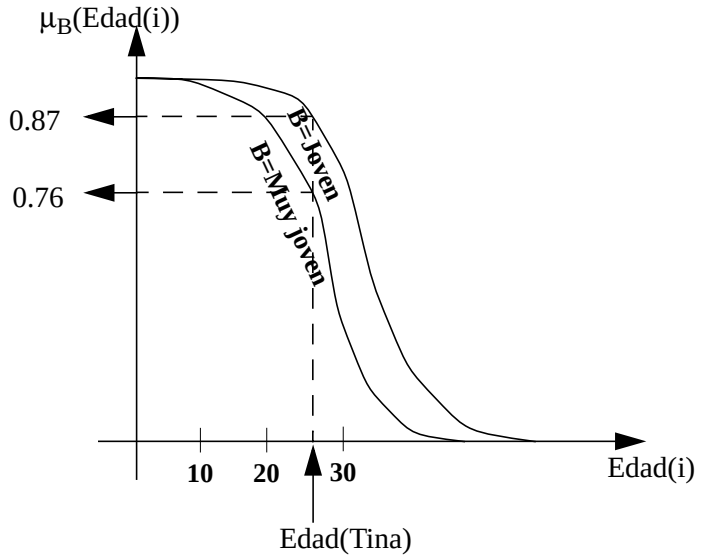

Figura A-2: Grado de verdad de una proposición borrosa

Una proposición borrosa sin calificador puede considerarse como un caso particular de proposición borrosa calificada, en el que el calificador corresponde a la etiqueta "verdadero", representada por la función S igual a la identidad.

A.4.2.2 Proposición borrosa con calificador borroso de probabilidad

La forma canónica de una proposición borrosa p de este tipo es la siguiente:

p: Pro(*X* es $\underline{B}$) es P [EC. A.37]

donde *X* es una variable que toma valores x en un conjunto universal U, $\underline{B}$ es un subconjunto borroso definido en U, P es un calificador borroso de probabilidad, y Pro(*X* es $\underline{B}$) es la probabilidad del del evento borroso "*X* es $\underline{B}$".

Las proposiciones borrosas de este tipo describen restricciones en las distribuciones de probabilidad en el universo U. Para una distribución de probabilidad f en U, calcularemos la probabilidad como:

$$\text{Pro}(X \text{ es } \underline{B}) = \sum_{x \in U} f(x) \cdot \mu_{\underline{B}}(x) \qquad \text{[EC. A.38]}$$

y el grado de verdad E(p) de la proposición p, con el calificador P, se calculará aplicando la función que representa P sobre la probabilidad de la proposición sin calificador:

$$E(p) = P\left(\sum_{x \in U} f(x) \cdot \mu_{\underline{B}}(x)\right) \qquad \text{[EC. A.39]}$$

Los posibles valores de P se representan con etiquetas de la forma: "probable", "muy probable", "improbable", etc. cuyo significado viene dado por curvas como las representadas en la figura A-3

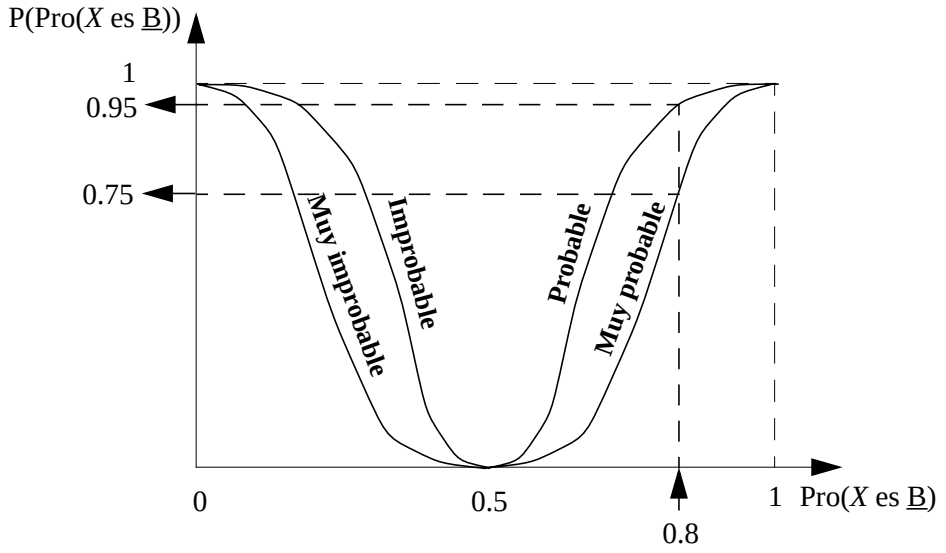

Figura A-3: Ejemplo de calificadores de probabilidad borrosos

Por ejemplo, supongamos una variable *T* que representa la temperatura media diaria en ºC de una ciudad concreta en un mes del año. Supongamos una proposición de la forma "la temperatura T está alrededor de 20 ºC", y que conocemos la definición del conjunto borroso "alrededor de 20 ºC" y la función f de densidad de probabilidad para los diferentes valores de *T*; entonces podemos calcular la probabilidad de esta proposición aplicando la fórmula [EC. A.38]; supongamos que es 0.8. Aplicando diferentes calificadores de probabilidad se obtienen proposiciones con diferentes valores de verdad; por ejemplo, para p = "es probable que la temperatura T esté alrededor de 20 ºC" el grado de verdad será 0.95 (ver figura A-3), pero para la proposición "es muy probable que la temperatura T esté alrededor de 20 ºC" será sólo 0.75.

A.4.3 Proposiciones borrosas condicionales y no calificadas

Una proposición p de este tipo se expresa por la forma canónica:

$$p: \text{Si } X \text{ es } \underline{A}, \text{ entonces } Y \text{ es } \underline{B} \qquad \text{[EC. A.40]}$$

donde *X* y *Y* son variables definidas en conjuntos universales U_X y U_Y respectivamente, y $\underline{A}$ y $\underline{B}$ son conjuntos borrosos en U_X y U_Y respectivamente. También se suelen representar con el símbolo "$\rightarrow$" del condicional: $\underline{A} \rightarrow \underline{B}$.

Estas proposiciones también pueden verse como proposiciones de la forma:

$$p: \langle X, Y \rangle \text{ es R} \qquad \text{[EC. A.41]}$$

donde R es una relación borrosa (subconjunto borroso de $U_X \times U_Y$), definida como ([Fernández y Sáez-Vacas, 95])[1]:

$$\mu_R(x, y) = \max(\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x)) \qquad \text{[EC. A.42]}$$

1. Son posibles otras definiciones, por ejemplo, aplicando las leyes de Lukasiewicz ([Klir y Yuan, 95]):
$$\mu_R(x, y) = \min(1, 1 - \mu_A(x) + \mu_B(y))$$

equivalente la sentencia $\neg\underline{A} \vee (\underline{A} \wedge \underline{B})$.

A.4.4 Proposiciones borrosas condicionales y calificadas

Las proposiciones de este tipo se caracterizan por las siguientes formas canónicas:

p: Si *X* es $\underline{A}$, entonces *Y* es $\underline{B}$ es V [EC. A.43]

donde V es un calificador del grado de verdad, o bien:

p: Pro(*X* es $\underline{A}$ | *Y* es $\underline{B}$) es P [EC. A.44]

siendo P un calificador de la probabilidad condicionada Pro(*X* es $\underline{A}$ | *Y* es $\underline{B}$).

A.5 Etiquetas lingüísticas

Las etiquetas lingüísticas son adverbios de cantidad, como “muy”, “bastante”, “más o menos”, etc., que se usan dentro de sentencias borrosas como modificadores de predicados borrosos, valores de verdad borrosos o probabilidades borrosas. Por ejemplo, la proposición “x es joven”, cuyo significado se interpreta como “x es joven es verdadero”, se puede modificar con la etiqueta “muy” de tres formas diferentes:

“x es muy joven es verdadero”

“x es joven es muy verdadero”

“x es muy joven es muy verdadero”

De forma similar, la sentencia “x es joven es probable” puede modificarse como “x es joven es muy probable”, etc.

En general, una proposición borrosa de la forma p: x es $\underline{B}$, y una etiqueta lingüística H, se puede construir una proposición modificada,

Hp: x es H$\underline{B}$ [EC. A.45]

donde H$\underline{B}$ representa el predicado borroso obtenido al aplicar la etiqueta H al predicado original $\underline{B}$. Otras modificaciones posibles resultan al aplicar H sobre el grado de verdad (equivalente a modificar el predicado borroso, apartado A.4.2.1) o el valor de probabilidad utilizados en la proposición dada (apartado A.4.2.2).

Las etiquetas lingüísticas sólo se aplican en lógica borrosa, ya que en la clásica lógica booleana no tienen interpretación posible sentencias como “muy horizontal”, “muy rectángulo”, etc.

Cualquier etiqueta lingüística H puede interpretarse como una función h, que se denomina *modificador*, definida en el intervalo unidad [0, 1]:

$h: [0, 1] \rightarrow [0, 1]$ [EC. A.46]

Dado un predicado borroso $\underline{B}$, el significado del predicado H$\underline{B}$ modificado se obtendrá aplicando la función h, asociada a la etiqueta H, sobre el grado de verdad del predicado sin modificador:

$$\mu_{HB}(x) = h(\mu_B(x)) \quad \text{[EC. A.47]}$$

Se denominan *modificadores fuertes* a aquéllos en los que *h(a) < a*, para todo *a* perteneciente al intervalo [0, 1], mientras que los *débiles* son aquéllos en los que *h(a) > a*. El modificador

identidad es aquél en el que *h(a) = a*. Cualquier modificador debe cumplir las siguientes propiedades ([Klir y Yuan, 95]):

1. h(0) = 0 y h(1) = 1
2. h es una función continua
3. si h es fuerte, entonces h^{-1} es débil
4. dado otro modificador g, la composición de g con h y de h con g son también modificadores; además, si ambos son fuertes (o débiles), las composiciones serán también fuertes (o débiles).

Generalmente se utiliza la siguiente clase de funciones

$$h_\alpha(a) = a^\alpha,\ a \in [0, 1],\ \alpha \in \Re^+ \qquad \text{[EC. A.48]}$$

para definir los modificadores asociados a las etiquetas lingüísticas. Cuando $\alpha < 1$, h_α es un modificador débil; cuando $\alpha > 1$, h_α es fuerte; y h_1 es el modificador identidad. Algunas definiciones de modificadores podrían ser ([Zadeh, 73]):

muy(a) = a^2

sumamente(a) = a^3

etc.

Apéndice B:

DESARROLLO DE ALGUNAS EXPRESIONES MATEMÁTICAS

B.1 Relación entre consistencia y k-consistencia borrosas

En este apartado demostraremos que las expresiones:

[EC. 4.29]: C es consistente sobre T $\Leftrightarrow (\forall t \in T)\ (SB(t) \geq \mu_{T_C(C)}(t))$

[EC. 4.30]: C es k-consistente sobre T $\Leftrightarrow \left|\underline{T}_C(C) \cap \underline{P}\right| \geq k \cdot \left|\underline{T}_C(C)\right|$, para $k \in [0, 1]$

definidas en el apartado 4.3.2.6, son equivalentes entre sí para el valor k=1. Además, la segunda es una generalización de la primera para cualquier k (es decir, toda cláusula consistente es también k-consistente).

- Demostración "toda cláusula consistente es k-consistente (la condición de k-consistencia es una generalización de la consistencia)": [EC. 4.29] $\Rightarrow$ [EC. 4.30], para cualquier valor $k \in [0, 1]$

Partiendo de la [EC. 4.29] y aplicando la función

$$f(x) = \min(\mu_{T_C(C)}(t), x)$$

a ambos lados de la desigualdad no se altera el sentido de la misma:

$$(\forall t \in T)\ (\min(\mu_{T_C(C)}(t), SB(t)) \geq \min(\mu_{T_C(C)}(t), \mu_{T_C(C)}(t)))$$

El parámetro k toma valores en el intervalo [0, 1], por tanto, se puede introducir como factor en la parte derecha de la desigualdad:

$$(\forall t \in T)\ (\min(\mu_{T_C(C)}(t), SB(t)) \geq k \cdot \mu_{T_C(C)}(t))$$

Tomando el sumatorio en las tuplas en que se cumple ($\forall t \in T$) se llega a:

$$\sum_{t \in T} \min\left(\mu_{T_C(C)}(t), SB(t)\right) \geq k \cdot \sum_{t \in T} \left(\mu_{T_C(C)}(t)\right) \Leftrightarrow$$

$$\Leftrightarrow \quad \left|\underline{T_C}(C) \cap \underline{P}\right| \geq k \cdot \left|\underline{T_C}(C)\right|$$

que es exactamente la expresión [EC. 4.30].

- Demostración “toda cláusula k-consistente para k=1 cumple la condición de consistencia”: [EC. 4.29] $\Leftarrow$ [EC. 4.30], cuando k = 1

Partiendo de [EC. 4.30], se puede expresar la cardinalidad escalar en forma de sumatorio, y sustituir el parámetro k = 1:

$$\sum_{t \in T} \min\left(\mu_{T_C(C)}(t), SB(t)\right) \geq \sum_{t \in T} \left(\mu_{T_C(C)}(t)\right)$$

Supongamos ahora que el conjunto T se parte en dos subconjuntos del siguiente modo:

$$T1 = \{\, t \in T \mid SB(t) \geq \mu_{T_C(C)}(t) \,\}$$

$$T2 = \{\, t \in T \mid SB(t) < \mu_{T_C(C)}(t) \,\}$$

en ese caso, la condición anterior queda:

$$\sum_{t \in T1} \mu_{T_C(C)}(t) + \sum_{t \in T2} SB(t) \geq \sum_{t \in T} \left(\mu_{T_C(C)}(t)\right.$$

$$\Rightarrow \quad \sum_{t \in T2} SB(t) \geq \sum_{t \in T2} \left(\mu_{T_C(C)}(t)\right.$$

que sólo puede cumplirse cuando T2 es el conjunto vacío. Por tanto, todas las tuplas de T deben cumplir la expresión [EC. 4.29]:

$$(\forall t \in T)\,(SB(t) \geq \mu_{T_C(C)}(t))$$

B.2 Cálculo de *NumLit(m,u)*

Denominamos:

NumLit(m,u) al número de literales afirmativos distintos, construidos a partir de un predicado q_m de grado *m*, que son candidatos a ser el siguiente antecedente de una cláusula en construcción con *u* variables usadas.

Todos los literales candidatos deben tener, al menos, una variable usada, por tanto, si se denomina:

NumLitJ(m,u) al número de literales afirmativos distintos con variables usadas en J términos, construidos a partir de un predicado q_m de grado *m*, que son candidatos a ser el siguiente antecedente de una cláusula en construcción con *u* variables usadas.

se cumple que:

$$\text{NumLit}(m, u) = \sum_{J=1}^{m} \text{NumLitJ}(m, u) \qquad \text{[EC. B.1]}$$

En el apartado B.2.1 se calcula la expresión de NumLitJ(m,u) que, sustituida en la ecuación anterior, da lugar a la expresión:

$$\text{NumLit}(m, u) = \sum_{j=1}^{m} \binom{m}{j} \cdot u^{j} \cdot \text{anynew}(m - j) \qquad \text{[EC. B.2]}$$

donde:

anynew(i) = número de diferentes combinaciones de variables nuevas para una tupla con *i* términos; este valor se calcula en el apartado B.2.2

B.2.1 Cálculo de *NumLitJ(m,u)*

Si denominamos:

m = grado de un predicado q_m asociado a una relación Q_m de la base de datos

u = número de variables usadas en una cláusula en construcción

J = número de términos de un literal que son variables usadas (en una cláusula en construcción)

NumLitJ(m,u) = número de literales afirmativos distintos con variables usadas en J términos, construidos a partir de un predicado q_m de grado *m*, que son candidatos a ser el siguiente antecedente de una cláusula en construcción con *u* variables usadas.

entonces, el número de literales afirmativos de nombre q_m con exáctamente J términos que son variables usadas (aunque pueden aparecer repetidas) será:

$$\text{NumLitJ}(m, u) = \binom{m}{J} \cdot u^{J} \cdot \text{anynew}(m - J) \qquad \text{[EC. B.3]}$$

donde:

anynew(i) = número de diferentes combinaciones de variables nuevas para una tupla con *i* términos; este valor se calcula en el apartado B.2.2

El primer valor de NumLitJ(m,u) será:

$$\text{NumLit1}(m,u) = m \cdot u \cdot \text{anynew}(m - 1) \qquad \text{[EC. B.4]}$$

B.2.2 Cálculo de *anynew(i)*

Denominamos:

anynew(i) = número de diferentes combinaciones de variables nuevas para una tupla con *i* términos

este valor depende únicamente del tamaño *i* del literal a considerar (número de términos). Por ejemplo, para una tupla de dos términos sólo son posibles dos combinaciones diferentes de variables nuevas, que son:

(V_1, V_1) = ambos términos son variables nuevas y, además, iguales entre sí

(V_1, V_2) = ambos términos son variables nuevas y, además, distintas entre sí

ya que otras combinaciones son repetición de las anteriores:

(V_2, V_2) = ambos términos son variables nuevas y, además, iguales entre sí, igual que (V_1, V_1)

(V_2, V_1) = ambos términos son variables nuevas y, además, iguales entre sí, igual que (V_1, V_2)

(V_2, V_3) = ambos términos son variables nuevas y, además, iguales entre sí, igual que (V_1, V_2)

etc.

Por tanto, *anynew(2) = 2*. Otros valores de anynew(i) se pueden calcular con la expresión siguiente:

$$\text{anynew}(i) = \sum_{j=0}^{i} \text{new}(i, j) \qquad \text{[EC. B.5]}$$

donde new(i,j) es el número de diferentes combinaciones de i términos con j variables nuevas (que pueden repetirse):

$$\text{new}(i, 0) = \begin{cases} 1, \text{ si } i = 0 \\ 0, \text{ si } i > 0 \end{cases} \qquad \text{[EC. B.6]}$$

$$\text{new}(i, j) = \begin{cases} 0, \text{ si } i < j \\ \text{new}(i-1, j-1) + j \cdot \text{new}(i-1, j), \text{ si } i \geq j \end{cases} \qquad \text{[EC. B.7]}$$

B.3 Estimación del número medio de tuplas expandidas con las que debe evaluarse un literal

Sea un conjunto con N tuplas, todas expandidas a partir de una única tupla original. Supongamos que M de ellas satisfacen un literal L_i candidato. ¿Con cuántas habrá que evaluar, en media, hasta encontrar una de estas M tuplas?

- La probabilidad de que la primera tupla elegida sea una de las M que satisfacen el literal L_i será:

$$P_1 = \frac{M}{N}$$

- La probabilidad de que haya que evaluar dos tuplas será la probabilidad de que la primera no sea ninguna de las M y la segunda sí lo sea:

$$P_2 = \frac{N - M}{N} \cdot \frac{M}{N - 1}$$

- La probabilidad de que haya que evaluar tres tuplas será la probabilidad de que la primera no sea ninguna de las M, la segunda tampoco y la tercera sí lo sea:

$$P_3 = \frac{N - M}{N} \cdot \frac{N - M - 1}{N - 1} \cdot \frac{M}{N - 2}$$

- En general, la probabilidad de que haya que evaluar *n* tuplas hasta encontrar una de las M que satisfacen al literal será:

$$P_n = \frac{N - M}{N} \cdot \frac{N - M - 1}{N - 1} \cdot \frac{N - M - 2}{N - 2} \cdot \ldots \cdot \frac{N - M - n + 2}{N - n + 2} \cdot \frac{M}{N - n + 1}$$

Por tanto, se puede calcular el número medio de tuplas con las que debe evaluarse L_i hasta encontrar una de las M que lo satisfacen:

$$NumEva = 1 \cdot P_1 + 2 \cdot P_2 + \ldots + n \cdot P_n$$

Sustituyendo los diferentes valores calculados para P_n se llega a la siguiente expresión:

$$NumEva = \sum_{n=1}^{N-M+1} n \cdot \frac{M}{N - n + 1} \cdot \frac{\binom{N - M}{n - 1}}{\binom{N}{n - 1}} \qquad \text{[EC. B.8]}$$

Se puede comprobar que esta ecuación se reduce a la siguiente, sin más que desarrollar el sumatorio y simplificar términos:

$$NumEva = \frac{N + 1}{M + 1} \qquad \text{[EC. B.9]}$$

Apéndice C:

MANUAL DE USUARIO Y DE INSTALACIÓN

C.1 Descripción

FZFOIL es un programa escrito en C++ basado en el sistema FOIL de Quinlan. Las principales características de FOIL son:

1. Los datos de entrada son relaciones almacenadas en un fichero de texto o directamente introducidas por el dispositivo de entrada estándar (teclado). Estos datos están expresados a través de un conjunto de tuplas para cada relación (formato extensional).

2. Induce una definición lógica, en forma de disyunción de cláusulas de Horn, para una o varias de las relaciones de entrada. Las definiciones se construyen mediante literales aplicados sobre variables, que pueden construirse a partir de las relaciones de entrada (incluyendo la relación objetivo en definiciones recursivas) y/o de relaciones predefinidas (*igualdad* / *desigualdad* / *mayor que* / *menor que*, entre dos variables o entre una variable y una constante). La salida se realiza por el dispositivo de salida estándar (pantalla).

3. Utiliza un heurístico de evaluación de literales basado en medidas de *ganancia* de información.

4. En ocasiones se permite la construcción de definiciones incompletas y/o inconsistentes.

Como novedades, FZFOIL incluye, además de las anteriores características:

1. Se permite la introducción de relaciones expresadas en forma intensional (además de las relaciones extensionales), mediante cláusulas de Horn construidas a partir de otras relaciones extensionales, intensionales o predefinidas.

2. Además, se permite definir relaciones borrosas, en las que sus tuplas tienen asignado un grado de pertenencia a la correspondiente relación.

3. Permite utilizar como heurísticos de evaluación la función *Ganancia* del FOIL original, así como medidas de Interés e Interés* (sobre conjuntos intermedios de tuplas, proyectados sobre el inicial).

4. Las definiciones lógicas que se inducen pueden ser borrosas (si utilizan literales borrosos). Además, se permite la utilización de etiquetas lingüísticas en los literales borrosos.

C.2 Opciones

La ejecución de FZFOIL se realiza mediante la orden 'fzfoil', que admite las siguientes opciones:

fzfoil [-n] [-N] [-v verb] [-V vars] [-a prec] [-c cons]
[-C comp] [-q cort] [-m maxt] [-s neg] [-k mins]
[-i fileIn] [-o fileOut]

El significado de estas opciones es el siguiente:

-n Los literales negados no se consideran. Esto será útil en dominios en los que no tenga sentido los literales negados.

-N Similar a la opción -n, pero permite la negación de los literales de igualdad: A≠B y A≠constante.

-v verb Determina el nivel de detalle de los mensajes a la salida: 0, 1, 2, 3, ó 4 (por defecto 1).

-V vars Establece el número máximo de variables que pueden usarse durante la construcción de una definición (por defecto 52).

-a prec Establece la precisión mínima de una cláusula (expresada en %). La precisión se calcula como la proporción de tuplas "+" respecto del total en el conjunto cubierto. No se permiten cláusulas con precisión inferior a *prec*. Valores posibles: enteros desde 0 hasta 100. Por defecto: 80.

-c cons Establece el valor del parámetro *k* para medir la k-consistencia de cláusulas y definiciones. Se consideran k-consistentes las cláusulas y definiciones en las que

$$N_C^+ \geq k \cdot (N_C^+ + N_C^-)$$

siendo N_C^+ y N_C^- el número de tuplas "+" y "-" cubiertas. Valores permitidos: reales de 0 a 1. Por defecto: 0.9

-C comp Establece el valor del parámetro *k* para medir la k-completitud de cláusulas y definiciones. Se consideran k-completas las cláusulas y definiciones en las que

$$N_C^+ \geq k \cdot (N_C^+ + M_C^+),$$

siendo N_C^+ el número de tuplas "+" cubiertas y M_C^+ el de tuplas "+" no cubiertas. Valores permitidos: reales de 0 a 1. Por defecto: 0.9

-q cobt Establece el valor del parámetro *k* de k-cobertura para determinar si una tupla es cubierta por una cláusula o si debe seguir en la relación residual. Se considera que se cumple la condición de k-cobertura para una tupla t sii:

$$\mu_{TC}(t) \geq k \cdot SB(t)$$

siendo $\mu_{TC}(t)$ el grado de pertenencia de t al conjunto cubierto y SB(t) el signo borroso de t. Valores permitidos: reales de 0 a 1. Por defecto: 0.8

-m maxt Establece el número máximo de tuplas para los conjuntos locales de entrenamiento. Por defecto 100000.

-s neg Obliga a que sólo se use una muestra aleatoria del conjunto inicial de tuplas negativas: *neg*% del total. Esto se usa cuando las relaciones objetivo tienen muchos atributos, pues en ese caso la suposición de mundo cerrado generaría demasiadas tuplas "-". Esta opción no tiene ningún efecto si se especifica el conjunto de tuplas negativas. Valores posibles: entre 0 y 100. Por defecto: 100.

-k mins Termina la ejecución del algoritmo después de *mins* minutos como máximo (si no se ha encontrado antes una definición consistente y completa).

-i fileIn Indica el nombre *fileIn* del fichero con los datos de entrada.

-o fileOut Indica el nombre *fileOut* del fichero en el que se escriben los resultados.

Las opciones pueden introducirse en la línea de órdenes (por defecto), o bien en el fichero ".fzfoil" (en ese caso, basta con ejecutar 'fzfoil' sin ninguna opción. Por ejemplo, la siguiente línea de orden (o un fichero ".fzfoil" que contenga esta línea):

```
fzfoil -v3 -i f1
```

ejecuta FZFOIL con nivel de detalle 3 y el fichero de entrada "f1", y escribe los mensajes de detalles de ejecución y la definición de salida directamente por la pantalla.

C.3 Formato del fichero de entrada

Los datos del fichero de entrada pueden agruparse en cuatro partes:

1. Especificación de dominios.
2. Definiciones extensionales de relaciones (conjuntos de tuplas).
3. Definiciones intensionales de relaciones.

4. Casos de prueba para las definiciones inducidas.

Toda línea precedida del carácter '|' se considera un comentario y su contenido se ignora.

C.3.1 Especificación de Dominios

Los dominios son los conjuntos de valores que puede tomar un atributo. Cada atributo dentro de una relación deberá pertenecer a uno (y sólo uno) de los dominios que se definan.

Los dominios pueden ser de tipo continuo o discreto:

a) Dominios continuos

Se definen mediante el nombre del dominio seguido por ": continuo." en la misma línea. Los valores de un dominio continuo son los números enteros y reales. Cualquier cadena de caracteres que pueda ser convertida a un *float*, con la función atof() de C, debería poder usarse como valor de dominio continuo en el atributo correspondiente de una tupla.

b) Dominios discretos

Se definen mediante el nombre del dominio seguido por ':' y a continuación la secuencia de valores que lo forman, separados por comas, y terminado con un '.' punto. La secuencia de valores puede introducirse en una o varia líneas.

Un valor de un dominio discreto puede repetirse en otros dominios discretos.

Hay tres clases de dominios discretos:

- Dominios ordenados (su nombre va precedido de '*').

  Los valores que pueden tomar tienen un orden natural dado por su secuencia de definición, apareciendo primero los más pequeños.

- Dominios desordenados (su nombre va precedido de '#').

  Sus valores no tienen ningún orden natural.

- Dominios posiblemente ordenados.

  En éstos, FZFOIL intentará descubrir si puede definirse algún orden entre los valores que lo forman. Esto puede ser útil a la hora de construir definiciones recursivas.

Los valores de un dominio pueden ser números o secuencias de caracteres (nombres). Ciertos caracteres (paréntesis, puntos, comas, punto y comas) sólo podrán aparecer en un nombre si van precedidos por el carácter de escape '\'.

En general, los valores de los dominios no se usarán para construir las definiciones lógicas. Sin embargo, se puede permitir cierto valor pueda aparecer dentro de un definición; para ello, se

deberá preceder su nombre por el carácter ‘*’ en el momento de especificar el dominio (a estos valores se les denomina “constantes de la teoría”).

Se debe tener en cuenta también que hay un carácter especial que puede considerarse valor de un dominio:

‘?’ para indicar un valor desconocido (ver [Cameron-Jones y Quinlan, 93b])

C.3.2 Definiciones extensionales de relaciones

Todas las relaciones extensionales se definen mediante un conjunto de tuplas (para las que la relación se satisface) y, opcionalmente, otro conjunto de tuplas para las que no se satisface. Es posible definir sólo el conjunto de tuplas perteneciente, y aplicar la suposición de mundo cerrado (*closed world assumption*), por la cual se considera que todas las demás no pertenecen a la relación.

La definición extensional de una relación consiste en una cabecera y uno o dos conjuntos de tuplas.

La cabecera puede especificarse de este modo:

[*]nombre(dominio, dominio, ..., dominio) <clave>/<clave>/.../<clave>

En esta cabecera se indica:

1. El nombre de la relación.
2. El nombre de los dominios para cada atributo.
3. Si se trata de una relación objetivo (es decir, para la que se busca una definición lógica) o no. Las relaciones objetivo carecerán del carácter ‘*’ precediendo a su nombre, mientras que las no objetivo lo tendrán.
4. Las claves, que limitan las formas en que puede utilizarse la relación. Constan de un carácter ‘#’ o ‘-’ para cada dominio. El carácter ‘#’ indica que el argumento correspondiente en la relación forma parte de una clave y, por lo tanto, debe estar ligado. El carácter ‘-’ indica que el argumento puede estar ligado o no. Si no se usan claves, se exploran todas las combinaciones posibles de argumentos.

Después de la cabecera se incluyen las tuplas que forman la relación y, opcionalmente, las que no pertenecen a la misma:

```
tupla perteneciente
tupla perteneciente
. . .
;                        |
tupla no perteneciente   | este conjunto es
tupla no perteneciente   | opcional
. . .                    |
```

Cada tupla consiste en una serie de valores constantes separados por comas, y debe escribirse en una única línea. El carácter ‘;’ separa el conjunto de tuplas pertenecientes de las no pertenecientes (en el caso en que éstas últimas se definan explícitamente).

Después de cada tupla, puede indicarse su grado de pertenencia a la relación, utilizando la expresión “:gp”, siendo gp un número real entre 0 y 1 (lógica borrosa) que indica su grado de pertenencia. Por defecto, se considera que éste es 1.

C.3.3 Definiciones intensionales de relaciones

Todas las relaciones intensionales se definen mediante una cláusula de Horn. La definición intensional se construye con literales basados en relaciones extensionales, en relaciones intensionales previamente definidas, o en relaciones predefinidas.

La introducción de relaciones definidas intensionalmente es opcional, siendo una extensión respecto de FOIL.

Cada relación intensional se define mediante una cabecera y una cláusula de Horn.

La cabecera puede especificarse de este modo:

[*]I_nombre(dominio, dominio, ..., dominio) <clave>/<clave>/.../<clave>

En esta cabecera se indica:

1. El nombre de la relación (precedido de “I_” para indicar que es intensional).
2. El nombre de los dominios para cada atributo.
3. Si se trata de una relación objetivo (es decir, para la que se busca una definición lógica) o no. Las relaciones objetivo carecerán del carácter ‘*’ precediendo a su nombre, mientras que las demás lo tendrán.
4. Las claves, que limitan las formas en que puede utilizarse la relación. Constan de un carácter ‘#’ o ‘-’ para cada dominio, con el mismo significado que se explicó al describir la definición extensional de relaciones.

Después de la cabecera se incluye la definición intensional, en forma de cláusula de Horn, terminado en punto ‘.’:

```
nombre(A,B...):-
rel1(A,...), rel2(C,...),
reln(B,...).
```

C.3.4 Relaciones predefinidas

Además de las relaciones definidas extensional o intensionalmente, se consideran también las siguientes relaciones predefinidas:

"=_const"	para igualdad entre una variable y un valor constante
"=_var"	para igualdad entre dos variables
">_const"	para comparación entre variable y constante
">_var"	para comparación entre dos variables

Las relaciones predefinidas pueden formar parte de la definición inducida, y también pueden utilizarse en la definición intensional de alguna relación. En cualquier caso, no es necesario definirlas en el fichero de entrada.

C.3.5 Casos de prueba para las definiciones inducidas

Los casos de prueba, que son opcionales, sirven para validar las definiciones lógicas que se induzcan. La definición de los casos de prueba se realiza del siguiente modo:

```
nombre de relación objetivo
tuplas de prueba
.
nombre de relación
tuplas de prueba
.
. . .
```

Cada tupla para prueba consiste en una tupla seguida por ": gp", siendo gp un número real entre 0 y 1 (lógica borrosa) que indica el grado de pertenencia a la relación. En caso de relaciones ordinarias, los únicos valores posibles para gp son 0 (tuplas negativas) y 1 (tuplas positivas).

C.4 Instalación

La última versión de FZFOIL está disponible, bajo licencia GNU, en la siguiente dirección URL:

ftp://ftp.gsi.dit.upm.es/public/fzfoil/fzfoil_*.tar.gz

Contiene los ficheros fuente del programa (archivados con *tar* y comprimidos con *gzip*). Basta descomprimirlo mediante las órdenes:

```
gzip -d fzfoil_*.tar.gz
tar xvf fzfoil_*.tar
```

con lo que se crea el directorio *fzfoil* y dentro de él los subdirectorios *source*, *include*, *obj* y *bin*. En los directorios *include* y *source* se instalan los ficheros fuente.

Dentro del subdirectorio *source* está el fichero *Makefile*, con el que se puede compilar para generar el ejecutable (fichero *fzfoil* en subdirectorio *bin*). Antes habrá que editar el fichero *Makefile* para seleccionar el compilador adecuado, según la versión de nuestro sistema operativo. Esta versión de FZFOIL ha sido probada en los sistemas operativos: SunOS 4.1.3, Solaris 2.x y Linux 2.x

GLOSARIO DE ACRÓNIMOS Y SÍMBOLOS

BD	Base de Datos
CDTI	Centro para el Desarrollo Tecnológico Industrial
DIT	Departamento de Ingeniería de sistemas Telemáticos
FOIL	Aprendiz inductivo de primer orden (*First Order Inductive Learner*)
FZFOIL	Aprendiz inductivo de primer orden borroso (FuZzy *First Order Inductive Learner*)
FTP	*File Transfer Protocol*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
ILP	Programación lógica inductiva (*Inductive Logic Programming*)
KDD	Descubrimiento de conocimiento en bases de datos (*Knowledge Discovery on Databases*)
LDE	Longitud de Descripción Extensional
LDI	Longitud de Descripción Intensional
PASO	Plan de Acción del SOftware
PIC	Puesto de Información Ciudadana
SEIC	SErvicio de Información Ciudadana
SIT	Sistema de Información de Transportes
URL	*Uniform Resource Locator*

$\neg$	Signo de negación lógica
$<$	Menor que
$\leq$	Menor o igual que
$\geq$	Mayor o igual que
$\vee$	Condición lógica OR
$\wedge$	Condición lógica AND
$\Rightarrow, \rightarrow$	Implicación lógica (si condicional)

$\Leftrightarrow, \leftrightarrow$	Doble implicación lógica (si y sólo si)
$\forall$	Cuantificador universal (para todo)
$\exists$	Cuantificador existencial (existe)
{x, y,...}	Conjunto de elementos x, y,...
{x | p(x)}	Conjunto de elementos definido por la propiedad p
χ_A	Función característica para el conjunto ordinario A
$x \in A$	Pertenencia de x al conjunto ordinario A
$x \notin A$	No pertenencia de x al conjunto ordinario A
μ_A	Función de pertenencia para el conjunto borroso $\underline{A}$
M	Conjunto de pertenencia (donde toma valores la función de pertenencia)
$\mu_A(x)$	Grado de pertenencia de x al conjunto borroso $\underline{A}$
$A = B$	Igualdad entre los conjuntos A y B
$A \neq B$	Desigualdad entre los conjuntos A y B
$A \cup B$	Unión entre los conjuntos A y B
$A \cap B$	Intersección entre los conjuntos A y B
$A - B$	Diferencia entre los conjuntos A y B
$A \subseteq B$	Relación de inclusión entre los conjuntos A y B
$A \subset B$	Relación de inclusión estricta entre los conjuntos A y B
$A \not\subset B$	Relación de no inclusión estricta entre los conjuntos A y B
$A \times B$	Producto cartesiano de A por B
$\varnothing$	Conjunto vacío
$\Re^+$	Conjunto de números reales positivos
U	Conjunto universal
U^n	Conjunto universal de dimensión n
$P(U)$	Conjunto de todos los subconjuntos ordinarios de U (conjunto potencia)
$\lvert T \rvert$	Cardinalidad del conjunto T (cardinalidad escalar para T borroso)
$h(\underline{A})$	Altura del conjunto borroso $\underline{A}$
${}^{\alpha}A$	Corte-alpha de valor α en el conjunto borroso A
${}^{[\alpha]}Q$	Conjunto ordinario con las tuplas cuyo grado de pertenencia a la relación borrosa Q es exactamente α
$\Lambda(\underline{A})$	Conjunto de niveles de un conjunto borroso $\underline{A}$

$\binom{N}{p}$	Combinación de N elementos tomados de p en p
$\log_2(n)$	Logaritmo en base 2 de n
$n!$	Factorial de n
$\sum x_i$	Sumatorio de elementos x_i
$\max(n,m)$	Máximo de n y m
$\min(n,m)$	Mínimo de n y m
$<a_1,..., a_m>$	Tupla formada por m valores: $a_1,..., a_m$
$+ <a_1,..., a_m>$	Tupla positiva formada por m valores: $a_1,..., a_m$
$- <a_1,..., a_m>$	Tupla negativa formada por m valores: $a_1,..., a_m$
$<t, \mu>$	Tupla borrosa t con grado de pertenencia μ a un conjunto T
SB(t)	Signo borroso de una tupla t
$t \in Q$	Pertenencia de una tupla t a la relación ordinaria Q
$\mu_Q(t)$	Grado de pertenencia de una tupla t a la relación borrosa Q
Q	Relación Q
$\underline{Q}$	Relación borrosa $\underline{Q}$
q(X,Y,...)	Literal construido con el predicado q (asociado a la relación Q), aplicado sobre las variables X, Y,...
L_i	Literal i-ésimo de una cláusula
HL	Literal borroso L modificado por la etiqueta lingüística H
L_0 :- L_1, L_2, ...	Cláusula de Horn con cuerpo: $L_1 \wedge L_2 \wedge$ ... y cabeza: L_0
C_i	Cláusula i-ésima de una definición
C^n	Cláusula con n literales en el cuerpo
D	Definición
T	Conjunto inicial de entrenamiento
T^+	Subconjunto de tuplas positivas de un conjunto T (para T borroso son las tuplas cuyo signo borroso es 1)
T^-	Subconjunto de tuplas negativas de un conjunto T (para T borroso son las tuplas cuyo signo borroso es 0)
$T^\sim$	Subconjunto de tuplas de T cuyo signo borroso no es 1 ni 0
T_i	Conjunto local de entrenamiento i-ésimo para literal L_i
$T_i^{[j]}$	Subconjunto de tuplas de T_i que tienen alguna extensión en T_j (siendo j > i)

$T_C(C)$	Conjunto cubierto por la cláusula C (también aplicable a una definición D)
$T_R(C)$	Relación residual asociada a cláusula C (también aplicable a una definición D)
N	Cardinalidad del conjunto T (es decir, $\|T\|$)
N_i	Cardinalidad del conjunto T_i (es decir, $\|T_i\|$)
N_i^+	Cardinalidad del conjunto T_i^+ (número de tuplas + de T_i)
N_i^-	Cardinalidad del conjunto T_i^- (número de tuplas – de T_i)
N_i^{++}	Número de tuplas del conjunto T_i^+ que satisfacen literal L_i
N_i^{-+}	Número de tuplas del conjunto T_i^- que satisfacen literal L_i
N_A	Número de hechos que satisfacen la condición lógica A
$N_{A\wedge B}$	Número de hechos que satisfacen A y B simultáneamente
$N_i^{[j]}$	Cardinalidad del conjunto $T_i^{[j]}$
$N_i^{[j]+}$	Cardinalidad del conjunto $T_i^{[j]+}$
$N_i^{[j]-}$	Cardinalidad del conjunto $T_i^{[j]-}$
i(x)	Interpretación de x
$A(t_i)$	Asignación de variables para el término t_i
E(S, A, i)	Evaluación de la sentencia S para la asignación de variables A y la interpretación i
$\models_{iA}(S)$	La interpretación i conjuntamente con la asignación de variables A satisface a la sentencia S. En lógica borrosa representa el grado de satisfacción.
$\models_t(L)$	Tupla t satisface al literal L (también aplicable a una cláusula C). En lógica borrosa representa un grado de satisfacción.
$h(\models_t(L))$	Grado de satisfacción de una tupla t para un literal modificado HL
LDE(Q)	Longitud de descripción extensional de la relación Q (también aplicable a una cláusula C y a una definición D)
LDI(L)	Longitud de descripción intensional del literal L (también aplicable a una cláusula C y a una definición D)
$Info(T_i)$	Información asociada al conjunto T_i
$Ganancia(L_i)$	Ganancia del literal L_i (parámetros medidos en conjunto T_i)
$Interes(L_i)$	Interés del literal L_i (parámetros medidos en conjunto T_i)
$Interes^*(L_i)$	Interés modificado del literal L_i (parámetros medidos en conjunto $T_1^{[i]}$)
$RI(A\rightarrow B)$	Interés (*rule interest*) de la regla lógica $A\rightarrow B$

Bel	Medida de credibilidad
Pl	Medida de plausibilidad
m	Asignación de probabilidad básica (en la teoría de la evidencia)
Pos	Medida de posibilidad
Nec	Medida de necesidad
Pro	Medida de probabilidad
Pro(B|A)	Probabilidad de B condicionada por A
CT	Coste teórico
CE	Coste de evaluación

BIBLIOGRAFÍA

[Agrawal et al., 93]

R. AGRAWAL, T. IMIELINSKI y A. SWAMI. "Database Mining: A Performance Perspective". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 914-925.

[Aha, 92]

D. W. AHA. "Relating Relational Learning Algorithms". En *Inductive Logic Programming*, STEPHEN MUGGLETON, Academic Press, San Diego (CA), 1992. Págs. 233-258.

[Aha, 95]

D. W. AHA. "Machine Learning". A tutorial presented at the Fifth International Workshop on Artificial Intelligence & Statistics. En *URL http://www.aic.nrl.navy.mil/~aha/slides.html/ml-tutorial.ps*, Enero 1995.

[Aha, 95b]

D. W. AHA. "Machine Learning: An Annotated Bibliography for the 1995 AI & Statistics Tutorial on Machine Learning". En *URL http://www.aic.nrl.navy.mil/~aha/slides.html/ml-tutorial-bib.ps*, Enero 1995.

[Aha et al., 91]

D. W. AHA, D. KIBLER y M. K. ALBERT. “Instance-Based Learning Algorithms”. En *Machine Learning*, vol. 6, 1991. Págs. 37-66.

[Ali y Pazzani, 93]

K. ALI, y M. PAZZANI. “HYDRA: A Noise-tolerant Relational Concept Learning Algorithm”. En *Proc. of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993. Págs. 1064-1070. (En *URL ftp://ftp.ics.uci.edu/pub/machine-learning-papers/publications/Ali-IJCAI- 93-Hydra.ps.Z*).

[Ali et al., 94]

K. ALI, C. BRUNK y M. PAZZANI. “On learning Multiple Descriptions of a Concept”. En *URL ftp://ftp.ics.uci.edu/pub/machine-learning-papers/publications/*

Ali-TAI-MultipleModels.ps.Z.

[Bell, 93]
D. A. BELL. "From Data Properties to Evidence". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 965-969.

[Berdagano, 93]
F. BERDAGANO. "Inductive Database Relation". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 969-972.

[Bezdek y Pal, 92]
J. C. BEZDEK y S. K. PAL. *Fuzzy Models for Pattern Recognition: Methods That Search for Structures in Data*, IEEE Press, New York, 1992.

[Botta et al., 92]
M. BOTTA, A. GIORDANA y L. SAITTA. "Comparison of Search Strategies in Learning Relations". En *Proceedings of ECAI'92*. John Wiley & Sons, 1992. Págs. 451-455.

[Botta y Giordana, 93]
M. BOTTA y A. GIORDANA. "SMART+: A Multi-Strategy Learning Tool". En *Proceedings of IJCAI'93*, vol. 2, 1993. Págs. 937-943.

[Brachman y Anand, 96]
R. J. BRACHMAN y T. ANAND. "The Process of Knowledge Discovery in Databases". En *Advances in Knowledge Discovery and Data Mining*, Fayyad, Piatetsy-Shapiro, Smyth y Uthurusamy Eds., AAAI Press, Menlo Park, California, 1996. Págs. 37-57.

[Briscoe y Caelli, 96]
G. BRISCOE y T. CAELLI. *A Compendium of Machine Learning, vol.1: Symbolic Machine Learning*, Ablex Publishing Corporation, Norwood, NJ, 1996.

[Brunk y Pazzani, 91]
C. A. BRUNK y M. J. PAZZANI. "An Investigation of Noise-Tolerant Relational Concept Learning Algorithms". En *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. Págs. 389-393.

[Cameron-Jones y Quinlan, 93]
R. M. CAMERON-JONES y J. R. QUINLAN. "Avoiding Pitfalls When Learning Recursive Theories". En *URL ftp://ftp.cs.su.oz.au/pub/ml/cj+q.ijcai93.ps*.

[Cameron-Jones y Quinlan, 93b]
R. M. CAMERON-JONES y J. R. QUINLAN. "First Order Learning, Zeroth Order Data". En *Proc. AI'93 Australian Joint Conference on Artificial Intelligence*, Melbourne, 1993. Págs. 316-321. (En *URL ftp://ftp.cs.su.oz.au/pub/tr/TR93_468.ps.Z*;

también en *URL ftp://ftp.cs.su.oz.au/pub/ml/cj+q.ai93.ps*).

[Cameron-Jones y Quinlan, 94]
R. M. CAMERON-JONES y J. R. QUINLAN. “Efficient top-down induction of logic programs”. En *SIGART*, vol. 5, 1994. Págs. 33-42. (En *URL ftp://ftp.cs.su.oz.au/pub/ml/cj+q.sigart.ps*).

[Carbonell et al., 84]
J. G. CARBONELL, R. S. MICHALSKI y T. M. MITCHELL. "An Overview of Machine Learning". En *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Tioga Publishing, Palo Alto, California, 1984. Págs. 3-21.

[Carbonell, 90]
J. G. CARBONELL. "Introduction: Paradigms for Machine Learning". *Machine Learning: paradigms and methods*, J. G. CARBONELL, MIT Press, 1990. Págs. 1-9.

[Carbonell y Langley, 87]
J. CARBONELL y P. LANGLEY. "Learning, Machine". En *Encyclopedia of Artificial Intelligence*, Stuart C. Shapiro, Ed. John Wiley & Sons, 1987. Págs. 464-488.

[Castro y Zurita, 95]
J. L. CASTRO y J. M. ZURITA. “An Inductive Learning Algorithm in Fuzzy Systems”. En *Technical Report 950116, DECSAI* (Dep. of Computer Science and Artificial Intelligence), E.T.S. Ingeniería Informática, Univ. Granada, Mayo 1995. (En *URL ftp://pirata.ugr.es/tech_reports/difuso/zurita/tr95116.ps.Z*).

[Catlett, 91]
J. CATLETT. “Megainduction: Machine Learning on Very Large Databases. Chapter 4: Subsets”. Ph. D. thesis, Basser Dep. of Computer Science, Univ. Sydney, 1991. En *URL ftp://ftp.cs.su.oz.au/pub/tr/TR91_402/TR91_402.ps.Z.*

[Cendrowska, 88]
J. CENDROWSKA. "PRISM: An algorithm for inducing modular rules". En *Knowledge-based systems*, vol. 1, 1988. Págs. 255-276.

[Clark y Niblett, 89]
P. CLARK y T. NIBLETT. “The CN2 Induction Algorithm”. En *Machine Learning*, vol. 3, 1989. Págs. 262-283.

[Cleary y Trigg, 95]
J. G. CLEARY y L. E. TRIGG. “K*: An Instance-based Learner Using an Entropic Distance Measure”. En *URL http://www.cs.waikato.ac.nz/~ml/publications/1995/Cleary95_KStar.ps.gz.*

[Codd, 70]
E. F. CODD. “A relational model for large shared data banks”. En *Communications of*

the ACM, vol. 13 (6), 1970. Págs. 377-389.

[Conklin et al., 93]
D. CONKLIN, S. FORTIER y J. GLASGOW. "Knowledge Discovery in Molecular Databases". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 985-987.

[Cook et al., 96]
D. J. COOK, L. B. HOLDER y S. DJOKO. "Scalable Discovery of Informative Structural Concepts Using Domain Knowledge". En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 59-68.

[Cox, 92]
E. COX. "Fuzzy fundamentals". En *IEEE Spectrum*, Octubre 1992. Págs. 58-61.

[Chiueh y Katz, 93]
T. CHIUEH y R. H. KATZ. "A History Approach of Automatic Relationships Establishment for VLSI Design Databases". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 987-990.

[Darling, 97]
C. B. DARLING. "Datamining for the masses". En *Datamation*, vol. 43(2), Febrero 1997. (En *URL http://www.datamation.com/PlugIn/issues/1997/feb/O2mineframe.html*).

[Date, 86]
C. J. DATE. *An Introduction to Database Systems*, vol 1. 4ª edición, Addison-Wesley, Reading, Mass, 1986.

[Davis, 85]
R. E. DAVIS. "Logic Programming and Prolog: A Tutorial". *IEEE Software*,Septiembre 1985. Págs 53-62.

[De Jong, 88]
K. A. DE JONG. "Learning with Genetic Algorithms: An Overview". En *Machine Learning*, vol. 3, 1988. Págs. 121-138.

[De Jong et al., 93]
K. A. DE JONG, W. M. SPEARS y D. F. GORDON. "Using Genetic Algorithms for Concept Learning". En *Machine Learning*, vol. 13, 1993. Págs. 161-188.

[Decker y Focardi, 95]
K. M. DECKER y S. FOCARDI. "Technology Overview: A Report on Data Mining". En *CSCS TR-95-02 Technical Report*, Swiss Scientific Computing Center, Mayo 1995. Págs. 1-29. (En *URL ftp://ftp.cscs.ch/pub/CSCS/techreports/1995/CSCS-TR-95-*

02.ps.Z).

[De Raedt et al., 93]
L. DE RAEDT, N. LAVRAC y S. DZEROSKI. "Multiple Predicate Learning". En *Proceedings of IJCAI'93*, vol. 2, 1993. Págs. 1037-1042.

[De Raedt y Bruynooghe, 93]
L. DE RAEDT y M. BRUYNOOGHE. "A Theory of Clausal Discovery". En *Proceedings of IJCAI'93*, vol. 2, 1993. Págs. 1058-1063.

[De Raedt y Muggleton, 93]
L. DE RAEDT y S. MUGGLETON. "Bibliography on Inductive Logic Programming". En *ftp://ftp.gmd.de/MachineLearning/ILP/public/bib/lrefs.bib.Z.*

[Dhar y Tuzhilin, 93]
V. DHAR y A. TUZHILIN. "Abstract-Driven Pattern Discovery in Databases". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 926-938.

[Dietterich y Michalski, 84]
T. G. DIETTERICH y R. S. MICHALSKI. "A Comparative Review of Selected Methods for Learning from Examples". *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Tioga Publishing, Palo Alto, California, 1984. Págs 41-81.

[DIG, 95]
DIG (Data Intelligent Group). "An Overview of Data Mining at Dun & Bradstreet". DIG White Paper 95/01. Sept. 1995. En *URL http://www.santafe.edu/~kurt/wp9501.shtml*.

[DIG, 95b]
DIG (Data Intelligent Group). "From Data Mining to Database Marketing". DIG White Paper 95/02. Oct. 1995. En *URL http://www.santafe.edu/~kurt/wp9502.shtml*.

[Dzeroski, 91]
S. DZEROSKI. *Handling Noise in Inductive Logic Programming*, Master's Thesis, University of Ljubljana, 1991.

[Dzeroski, 96]
S. DZEROSKI. "Inductive Logic Programming and Knowledge Discovery in Databases". En *Advances in Knowledge Discovery and Data Mining*, Fayyad, Piatetsy-Shapiro, Smyth y Uthurusamy Eds., AAAI Press, Menlo Park, California, 1996. Págs. 117-152.

[Dzeroski y Lavrac, 91]
S. DZEROSKI y N. LAVRAC. "Learning Relations from Noisy Examples: An Empirical Comparison of LINUS and FOIL". En *Proceedings of the Eighth*

International Workshop on Machine Learning, Evanston, IL, 1991. Págs. 399-402.

[Dzeroski y Lavrac, 92]
DZEROSKI y N. LAVRAC. "Refinement Graphs for FOIL and LINUS". En *Inductive Logic Programming*, S. MUGGLETON, Academic Press, San Diego (CA), 1992. Págs. 319-333.

[Dzeroski y Lavrac, 93]
S. DZEROSKI y N. LAVRAC. "Inductive Learning in Deductive Databases". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 939-949.

[Elkan, 93]
C. ELKAN. "The Paradoxical Success of Fuzzy Logic". En *Proceedings of the 11th National Conference on Artificial Intelligence AAAI'93*, MIT Press, 1993. Págs. 698-703. (En *URL ftp://cs.ucsd.edu/pub/paradoxicalsuccess.ps*).

[Elkan et al., 94]
C. ELKAN et al. "The Paradoxical Success of Fuzzy Logic and responses". En *Expert IEEE*, vol. 9(4), Agosto 1994. Págs. 3-49.

[Emde, 94]
W. EMDE. "Inductive Learning of Characteristic Concept Descriptions from Small Sets of Classified Examples". En *Proc. 7th European Conference on Machine Learning*, Berdagano y DeRaedt eds., Sprienger-Verlag, 1994. (En *URL ftp:// ftp.gmd.de/MachineLearning/GMD/papers/Emde-94-ECML-94.ps.Z*).

[Ezawa y Norton, 96]
K. J. EZAWA y S. W. NORTON. “Constructing Bayesian Networks to Predict Uncollectible Telecommunications Accounts”. En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 45-51.

[Fargier et al., 96]
H. FARGIER, J. LANG, R. MARTIN-CLOUAIRE, T. SCHIEX. “A Constraint Satisfaction Framework for Decision Under Uncertainty”. En *URL ftp://ftp.init.fr/ private/CSFDU.ps*.

[Fayyad, 96]
U. M. FAYYAD. “Data Mining and Knowledge Discovery: Making Sense Out of Data”. En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 20-25.

[Fayyad et al., 96]
U. M. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH y R. UTHURUSAMY. *Advances in Knowledge Discovery and Data Mining*. The AAAI Press, Menlo Park,

California, 1996.

[Fayyad et al., 96b]
U. M. FAYYAD, G. PIATETSKY-SHAPIRO y P. SMYTH. “From Data Mining to Knowledge Discovery”. En *Advances in Knowledge Discovery and Data Mining*, Fayyad, Piatetsy-Shapiro, Smyth y Uthurusamy Eds., AAAI Press, Menlo Park, California, 1996. Págs. 1-34.

[Feigenbaum, 92]
E. A. FEIGENBAUM. "Expert Systems: Principles and Practice". En *URL ftp:// ksl.stanford.edu/pub/KSL_Reports/KSL-91-79.ps*, 1992. Págs. 1-12. (This article will appear in *The Encyclopedia of Computer Science and Engineering*, 1992).

[Feller, 68]
W. FELLER. *An Introduction to Probability Theory and Its Applications*, vol 1. 3ª edición, John Wiley & Sons, New York, 1968.

[Fernández y Sáez-Vacas, 87]
G. FERNANDEZ y F. SAEZ-VACAS. *Fundamentos de Informática*, Alianza-Informática, Madrid, 1987.

[Fernández y Sáez-Vacas, 95]
G. FERNANDEZ y F. SAEZ-VACAS. *Fundamentos de Informática: Lógica, Autómatas, Algoritmos y Lenguajes*, Anaya Multimedia, Salamanca, 1995.

[Fisher, 96]
D. FISHER. “Iterative Optimization and Simplification of Hierarchical Clusterings”. En *URL ftp://ftp.mrg.dist.unige.it/pub/jair/pub/volume4/fisher96a.ps*. (En *Journal of Artificial Intelligence Research*, vol. 4, 1996. Págs. 147-179.

[Flach, 90]
P. A. FLACH. "Inductive characterisation of database relations". En *ITK ResearchReport* No. 23, November 1990. (En *URL ftp://ftp.gmd.de/MachineLearning/ ILP/public/papers/flach-ITKreport23.ps.Z*. Págs. 0-26). (En *Methodologies for Intelligent Systems*, 5. Ras, Zemankova y Emrich eds., Amsterdam, 1990. Págs. 371-378).

[Flach, 92]
P. A. FLACH. "Logical approaches to Machine Learning - an overview". En *Think*, vol. 1:2, September 1992. Págs. 25-36. (En *URL http://itkwww.kub.nl:2080/itk/Docs/ Think/1-2/peter.html*).

[Forsyth, 81]
R. FORSYTH. "BEAGLE: A Darwinian Approach to Pattern Recognition".

Kibernetes, vol. 10, Thales Publications, Great Britain, 1981. Págs 159-166.

[Frawley et al., 91]
W. J. FRAWLEY, G. PIATETSKI-SHAPIRO y C. J. MATHEUS. "Knowledge Discovery in Databases: An Overview". *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro y W. Frawley, AAAI-MIT Press, Menlo Park, California, 1991. Págs. 1-27.

[Frazier y Page, 93]
M. FRAZIER y C. D. PAGE. "Learnability in Inductive Logic Programming: Some Basic Results and Techniques". En *Proceedings of AAAI'93*, AAAI-MIT Press, Washington, DC, July 1993. Págs. 93-98.

[Freeman y Skapura, 93]
J. A. FREEMAN y D. M. SKAPURA. *Redes Neuronales: Algoritmos, aplicaciones y técnicas de propagación*, Addison-Wesley, 1993.

[Freitas y Lavington, 95]
A. A. FREITAS y S. H. LAVINGTON. "A data-parallel primitive for high-performance knowledge discovery in large databases". En *Internal Report CSM-242*, Department of Computer Science, University of Essex, Colchester (UK), Mayo 1995. (En *URL ftp://ftp.essex.ac.uk/pub/csc/technical-reports/CSM-242.ps.Z.* Paper submitted to the 12th International Conference on Data Engineering, New Orleans, Feb 1996)

[Gaines y Compton, 93]
B. R. GAINES y P. COMPTON. "Induction of Meta-Knowledge about Knowledge Discovery". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 990-992.

[Gallaire et al., 84]
H. GALLAIRE, J. MINKER y J. M. NICOLAS. "Logic and Databases: A Deductive Approach". *ACM Computing Surveys*, vol. 16(2), Junio 1984. Págs. 153-186.

[Genesereth y Nilsson., 87]
M. R. GENESERETH y N. J. NILSSON. *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[Ginsberg et al., 88]
A. GINSBERG, S. M. WEISS y P. POLITAKIS. "Automatic Knowledge Base Refinement for Classification Systems". En *Artificial Intelligence*, vol. 35, Elsevier Science Publishers, North-Holland, 1988. Págs. 197-226.

[Gómez y Fernández, 92]
A. J. GÓMEZ y G. FERNANDEZ. "Inducción de definiciones lógicas a partir de relaciones: mejoras en los heurísticos del sistema FOIL". En *Actas del Primer Congreso Nacional de Programación Declarativa PRODE'92*, Madrid, Septiembre

1992. Págs. 292-302.

[Gómez, 92]
A. J. GÓMEZ. *Sistema para Aprendizaje de Reglas en Bases de Datos Relacionales*, Proyecto Fin de Carrera, Dpto. Ing. Sistemas Telemáticos, E.T.S.I. Telecomunicación, Universidad Politécnica, Madrid, 1992.

[Gómez, 93]
A. J. GÓMEZ. "Inducción de reglas en bases de datos". En *Informática y automática*, vol. 26(3), 1993. Págs. 34-45.

[Gómez, 94]
A. J. GÓMEZ. "Induction of Rules in Databases: Modifications to the FOIL system". En *Proceedings of the 14th International Avignon Conference*, vol. 1, EC2, París, June 1994. Págs. 109-118.

[González, 89]
J. C. GONZÁLEZ. *Arquitectura para sistemas expertos con razonamiento aproximado*, Tesis Doctoral, Dpto. Ing. Sistemas Telemáticos, E.T.S.I. Telecomunicación, Universidad Politécnica, Madrid, 1989.

[González y Pérez, 95]
A. GONZÁLEZ y R. PÉREZ. "Completeness and Consistency Conditions for Learning Fuzzy Rules". En *URL ftp://pirata.ugr.es/tech_reports/difuso/gonzalez/tr95_103.ps.Z*. Dpto. de Ciencias de la Computación e Inteligencia Artificial, E.T.S Ingeniería Informática, Univ. Granada, 1995.

[Gray, 95]
J. N. GRAY. "Database Systems: A Textbook Case of Research Paying Off". En *URL http://www.cs.washington.edu/homes/lazowska/cra/database.html*.

[Grefenstette, 89]
J. GREFENSTETTE. "A System for Learning Control Strategies with Genetic Algorithms". En *ICGA'89*. Págs. 183-190.

[Gur-Ali y Wallace, 93]
O. GUR-ALI y W. A. WALLACE. "Induction of Rules Subject to a Quality Constraint: Probabilistic Inductive Learning". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 979-984.

[Han et al., 93]
J.HAN, Y.CAI y N.CERCONE. "Data-Driven Discovery of Quantitative Rules in Relational Databases". En *IEEE Transactions on knowledge and data engineering*, vol. 5(1), Febrero 1993. Págs. 29-40.

[Han et al., 94]
J. HAN, O. R. ZAÏANE y Y. FU. "Resource and Knowledge Discovery in Global

Information Systems: A Multiple Layered Database Approach". En *CMPT TR94-10 Technical Report*, Database Systems Laboratory, School of Computing Science, Simor Fraser University, Bulnaby, Canada, 1994. Págs. 1-30. (En *URL ftp://ftp.fas.sfu.ca/pub/cs/TR/1994/CMPT94-10.ps.Z*).

[Han et al., 97]
J. HAN, M. KAMBER y J. CHIANG. "Mining Multi-Dimensional Association Rules Using Data Cubes". En *CMPT TR97-06 Technical Report*, Database Systems Research Laboratory, School of Computing Science, Simor Fraser University, Bulnaby, Canada, 1997. Págs. 1-21. (En *URL ftp://ftp.fas.sfu.ca/pub/cs/TR/1997/CMPT97-06.ps.Z*).

[Hemerly et al., 93]
A. S. HEMERLY, M. A. CASANOVA y A. L. FURTADO. "Avoiding Misconstruals in Database Systems: A Default Logic Approach". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 994-996.

[Hernández y Stolfo, 95]
M. A. HERNÁNDEZ y S. J. STOLFO. "The Merge/Purge Problem for Large Databases". *Technical Report*, Dept. of Computer Science, Columbia University, New York. 1995.

[Hertz et al., 91]
J. HERTZ, A. KROGH y R. G. PALMER. *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

[Holder y Cook, 93]
L. B. HOLDER y D. J. COOK. "Discovery of Inexact Concepts from Structural Data". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 992-994.

[Holland, 92]
J. H. HOLLAND. "Algoritmos genéticos". En *Investigación y Ciencia*, Septiembre 1992. Págs. 38-45.

[Holsheimer y Kersten, 94]
M. HOLSHEIMER y M. L. KERSTEN. "Architectural Support for Data Mining". En *URL ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9429.ps.Z*, 1994. Págs. 1-12.

[Holsheimer y Siebes, 94]
M. HOLSHEIMER y A. SIEBES. "Data Mining: The Search for Knowledge in Databases". *URL ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9406.ps.Z*, 1994. Págs. 1-78.

[Horstkotte, 96]
E. HORSTKOTTE. “Fuzzy Expert Systems”. En *URL http://www.austinlinks.com/*

Fuzzy/expert_systems.html, 1996.

[Hume y Pazzani, 95]
T. HUME y M. J. PAZZANI. "Learning Sets of Related Concepts: A Shared Task Model". En *URL http://www.ics.uci.edu/~pazzani/CogSci95.html*, 1995. Págs. 1-11.

[Hurley, 92]
C. B. HURLEY. "An Architecture and other Key Results of Experimental Development of Network and Customer Administration Systems". En *The Management of Telecommunications Networks*, Ellis Horwood, 1992. Págs. 135-145.

[IBM, 95]
IBM BookManager BookServer. “4.0 Data Mining: Verification vs. Discovery”. En *URL http://booksrv2.raleigh.ibm.com:80/cgi-bin/bookmgr/bookmgr_cmd/BOOKS/datamine/4.0*.

[IEEE, 93]
"Special Issue on Learning and Discovery in Knowledge-Based Databases". *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), IEEE Computer Society, Diciembre 1993. Págs. 901-998.

[Janikow, 93]
C. Z. JANIKOW. “A Knowledge-Intensive Genetic Algorithm for Supervised Learning”. En *Machine Learning*, vol. 13, 1993. Págs. 189-228.

[Jiménez et al., 93]
A. JIMÉNEZ, R. GALÁN, R. SANZ, J. R. VELASCO. *Curso de control inteligente de procesos*, Publicaciones de la E.T.S.I. Industriales, Madrid, 1993.

[Jin et al., 95]
Y. JIN, J. JIANG y J. ZHU. "Neural Network Based Fuzzy Identification and Its Application to Modeling and Control of Complex Systems". En *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25(6), Junio 1995. Págs. 990-997.

[John et al., 96]
G. H. JOHN, P. MILLER y R. KERBER. “Stock Selection Using Rule Induction”. En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 52-58.

[Kantrowitz et al., 94]
M. KANTROWITZ, E. HORSTKOTTE y C. JOSLYN. “Answers to Frequently Asked Questions about Fuzzy Logic and Fuzzy Expert Systems”. En *URL http://www.cs.cmu.edu:8001/Web/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html*.

[Kijsirikul et al., 91]
B. KIJSIRIKUL, M. NUMAO y M. SHIMURA. "Efficient Learning of Logic Programs with Non-determinate, Non-discriminating Literals". En *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. Págs. 417-

421.

[Klir y Folger, 88]
G. J. KLIR y T. A. FOLGER. *Fuzzy sets, Uncertainty, and Information*. Prentice-Hall International, NY, 1988.

[Klir y Yuan, 95]
G. J. KLIR y B. YUAN. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall PTR, Upper Saddle River, NJ, 1995.

[Koczy, 96]
L. T. KOCZY. “Fuzzy If ... Then Rule Models and Their Transformation Into One Another”. En *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26(5), Septiembre 1996. Págs. 621-637.

[Kohavi et al., 94]
R. KOHAVI et al. "MLC++: A Machine Learning Library in C++". En *URL ftp:// starry.stanford.edu/pub/roonyk/mlc/mlctools.ps*, 1994. (Appears in Tools with AI'94).

[Lacruz et al., 93]
B. LACRUZ, P. LASALA y A. LEKUONA. "Generación Automática de Bases de Conocimiento a partir de Bases de Datos. Un caso de estudio". En *Actas de la CAEPIA'93*, AEPIA, Noviembre 1993. Págs. 446-453.

[Larousse, 96]
Gran Enciclopedia Larousse. Ed. Planeta, 1996.

[Le, 94]
T. V. LE. "Fuzzy Programming in Prolog". En *AI Expert*, vol. 9 (7), Julio 94. Págs. 32-36.

[Lee y Ong, 96]
H. Y. LEE y H. L. ONG. “Visualization Support for Data Mining”. En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 69-75.

[Lenat., 84]
D. B. LENAT. "The Role of Heuristics in Learning by Discovery: Three Case Studies". En *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Tioga Publishing, Palo Alto, California, 1984. Págs. 243-306.

[Leung y Lee, 88]
Y. Y. LEUNG y D. L. LEE. "Logic Approaches for Deductive Databases". *IEEE Expert*, Invierno 1988. Págs. 64-75.

[Limb y Meggs, 94]
P. R. LIMB y G. J. MEGGS. "Data mining, tools and techniques". En *BT Technology*

Journal, vol. 12(4), Octubre 1994. Págs. 32-41.

[Lippmann, 87]
R. P. LIPPMANN. “An Introduction to Computing with Neural Nets”. En *IEEE ASSP Magazine*, Abril 1987. Págs. 4-22.

[López de Mántaras, 91]
R. LÓPEZ DE MÁNTARAS. “A Distance-Based Attribute Selection Measure for Decision Tree Induction”. *Machine Learning*, vol. 6(1), 1991. Págs. 81-92.

[Mallen y Bramer, 95]
J. MALLEN y M. BRAMER. "CUPID: An Iterative Knowledge Discovery Framework". En *URL http://osiris.sis.port.ac.uk/technical-reports-index/kdpap.html*, University of Portmouth, UK, 1995.

[Mark, 96]
B. MARK. “Data Mining, Here We Go Again?”. En IEEE Expert, vol. 11(5), 1996. Págs. 18-19.

[Manning y Sparks, 92]
K. MANNING y E. SPARKS. "Service and Network Model Implementation". En *The Management of Telecommunications Networks*, Ellis Horwood, 1992. Págs. 97-108.

[Matheus et al., 93]
C. J. MATHEUS, P. K. CHAN y G. PIATETSKY-SHAPIRO. "Systems for Knowledge Discovery in Databases". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 903-913. (En *URL ftp://ceylon.gte.com/tkde-kdd.ps*, 1993. Págs. 0-21).

[Medina et al., 94]
J. M. MEDINA, O. PONS y M. A. VILA. “GEFRED. A Generalized Model of Fuzzy Relational Data Bases. Ver. 1.1”. En *URL ftp://decsai.ugr.es/pub/arai/tech_rep/medina/gefred.ps.Z*. Dpto. Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática, Univ. Granada.

[Mesrobian et al., 96]
E. MESROBIAN, R. MUNTZ, E. SHEK, S. NITTEL, M. LA ROUCHE, M. KRIGUER, C. MECHOSO, J. FARRARA, P. STOLORZ, H. NAKAMURA. “Mining Geophysical Data for Knowledge”. En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 34-44.

[Michalski, 84]
R. S. MICHALSKI. "A Theory and Methodology of Inductive Learning". *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell y T. M.

Mitchell, Tioga Publishing, Palo Alto, California, 1984. Págs. 83-134.

[Michalski, 86]
R. S. MICHALSKI. "Understanding the Nature of Learning: Issues and Research Directions". *Machine Learning: An Artificial Intelligence Intelligence Approach*, vol. 2, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Morgan Kaufmann Publishers, Los Altos, California, 1986. Págs. 3-25.

[Michalski, 87]
R. S. MICHALSKI. "Concept Learning". *Encyclopedia of Artificial Intelligence*, Stuart C. Shapiro, Ed. John Wiley & Sons, 1987. Págs. 185-194.

[Michalski et al., 84]
R. S. MICHALSKI, J. G. CARBONELL y T. M. MITCHELL. *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing, Palo Alto, California, 1984.

[Michalski et al., 86]
R. S. MICHALSKI, J. G. CARBONELL y T. M. MITCHELL. *Machine Learning: An Artificial Intelligence Approach*, vol. 2, Morgan Kaufmann Publishers, Los Altos, California, 1986.

[Mira et al., 95]
J. MIRA, A. E. DELGADO, J. G. BOTICARIO y F. J. DÍEZ. *Aspectos básicos de la Inteligencia Artificial*. Ed. Sanz y Torres, Madrid, 1995.

[Mooney y Califf, 95]
R. J. MOONEY y M. E. CALIFF. “Induction of First-Order Decision Lists: Results on Learning the Past Tense of English Verbs”. En *Journal of Artificial Intelligence Research*, vol. 3, 1995. Págs. 1-24. (En *URL http://www.cs.washington.edu/research/jair/volume3/mooney95a.ps*).

[Morik y Brockhausen, 97]
K. MORIK y P. BROCKHAUSEN. “A Multistrategy Approach to Relational Knowledge Discovery in Databases”. En *Machine Learning*, vol. 27, 1997. Págs. 287-312.

[Moxon, 96]
B. MOXON. “Defining Data Mining”. En *DBMS Data Warehouse Supplement (http://www.dbmsmag.com/9608d53.html)*, 1996.

[Muggleton, 92]
S. MUGGLETON. *Inductive Logic Programming*. Academic Press, San Diego (CA), 1992.

[Muggleton y Buntime, 88]
S. MUGGLETON y W. BUNTIME. “Machine Invention of First-Order Predicates by Inverting Resolution”. En *Proceedings of the First International Conference on*

Machine Learning, 1988. Págs. 339-352.

[Muggleton y Feng, 92]
S. MUGGLETON y C. FENG. "Efficient Induction of Logic Programs". En *Inductive Logic Programming*, S. MUGGLETON, Academic Press, San Diego (CA), 1992. Págs. 281-298.

[Muggleton y Michie, 96]
S. MUGGLETON y D. MICHIE. “Machine Intelligibility and the Duality Principle”. En *BT Technology Journal*, vol. 14(4), Oct. 96. Págs. 15-23.

[Murphy, 94]
P. M. MURPHY. “Utomated Rule-Base Creation via CLISP-INDUCE”. En *URL ftp://ftp.ics.uci.edu/pub/machine_learning_papers/publications/Murphy-CLISP-94-Induce.ps*.

[Neri y Saitta, 96]
F. NERI y L. SAITTA. “Exploring the Power of Genetic Search in Learning Symbolic Classifiers”. En *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18 (11), Nov. 1996. Págs. 1135-1141. (En *URL ftp://ftp.di.unito.it/pub/articles/GA/pami.ps.gz*).

[Nilsson, 82]
N. J. NILSSON. *Principles of Artificial Intelligence*. Springer-Verlag, Berlín, 1982. (Traducción española de J. Fernández-Biarge: *Principios de inteligencia artificial*, Díaz de Santos, Madrid, 1987).

[O´Leary, 95]
D. E. O´LEARY. "On the History of AI Applications, I: IEEE Expert, The First Nine Years". En *IEEE Expert*, vol. 10(1), Febrero 1995. Págs. 57-60.

[O´Leary, 95b]
D. E. O´LEARY. "On the History of AI Applications, II: IEEE Conference on Artificial Intelligence Applications". En *IEEE Expert*, vol. 10(1), Febrero 1995. Págs. 61-65.

[Oliver, 93]
J. R. OLIVER. "Discovering Individual Decision Rules: an Application of Genetic Algorithms". En *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo (CA), 1993. Págs. 216-222.

[Ozsu y Valduriez, 91]
M. T. OZSU y P. VALDURIEZ. *Principles of Distributed Database Systems*. 1ª edición, Prentice-Hall International, New Jersey, 1991.

[Pazzani et al., 91]
M. J. PAZZANI, C. A. BRUNK y G. SILVERSTEIN. "A knowledge-intensive

approach to learning relational concepts". En *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. Págs. 432-436.

[Pazzani y Brunk, 93]
M. PAZZANI y C. BRUNK. "Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning". En *Proceedings of AAAI'93*, AAAI-MIT Press, Washington, DC, July 1993. Págs. 328-334.

[Pazzani y Kibler, 92]
M. PAZZANI y D. KIBLER. "The Utility of Knowledge in Inductive Learning". *Machine Learning*, vol. 9, 1992. Págs. 57-94.

[Pednault, 91]
E. PEDNAULT. "Minimal-Length Encoding and Inductive Inference". En *Knowledge Discovery in Databases*, Ed. G. Piatetsky-Shapiro & W. J. Frawley, The AAAI Press, Menlo Park (CA), 1991. Págs. 71-92.

[Pérez, 96]
R. PÉREZ. "Aprendizaje de Conceptos y Algoritmos Genéticos". En *URL ftp:// decsai.ugr.es/pub/arai/tech_rep/fgr/tr_docto.ps.Z*. Technical Report DECSAI-96116. Dpto. Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática, Univ. Granada, Mayo 96.

[Piatetsky-Shapiro, 89]
G. PIATETSKY-SHAPIRO. "Discovery and Analysis of Strong Rules in Databases". En *Proceedings of Advanced Database System Symposium' 89*, Kyoto, Japón, Diciembre 1989. Págs. 135-142.

[Piatetsky-Shapiro, 91]
G. PIATETSKY-SHAPIRO. "Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop". *AI Magazine*, vol. 11(5), January 1991. Págs. 68-70.

[Piatetsky-Shapiro y Frawley, 91]
G. PIATETSKY-SHAPIRO y W. J. FRAWLEY. *Knowledge Discovery in Databases*. AAAI-MIT Press, Menlo Park, California, 1991.

[Piatetsky-Shapiro et al., 93]
G. PIATETSKY-SHAPIRO, C. MATHEUS, P. SMYTH y R. UTHURUSAMY. "KDD-93: Progress and Challenges in Knowledge Discovery in Databases". En *URL http://info.gte.com/~kdd/kdd-93-report.tex*, 1993. Págs. 1-7. (A long report on AAAI-93 KDD Workshop, submitted for publication).

[Plotkin, 69]
G. PLOTKIN. "A Note on Inductive Generalization". EN Machine Learning, vol. 5,

1969. Págs. 153-163.

[Pons et al., 94]
O. PONS, J. M. MEDINA, y M. A. VILA. “Inference from a Fuzzy Database using Fuzzy Rules”. En *URL ftp://decsai.ugr.es/pub/arai/tech_rep/opc/tr_deduce.ps.Z.* Technical Report DECSAI-94113. Dpto. Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática, Univ. Granada.

[Quinlan, 84]
J. R. QUINLAN. "Learning Efficient classification Procedures and Their Application to Chess End Games". En *Machine Learning: An Artificial Intelligence Intelligence Approach*, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Tioga Publishing, Palo Alto, California, 1984. Págs. 463-482.

[Quinlan, 86]
J. R. QUINLAN. "Induction of Decision Trees". *Machine Learning*, vol. 1, 1986. Págs. 81-106.

[Quinlan, 87]
J. R. QUINLAN. "Generating Production Rules From Decision Trees". En *Proceedings of IJCAI-87*, Milan, Italy, 1987. Págs. 304-307.

[Quinlan, 89]
J. R. QUINLAN. “Unknown Attribute Values in Induction” En *URL ftp:// ftp.cs.su.oz.au/pub/ml/q.ml89.ps.*

[Quinlan, 90]
J. R. QUINLAN. "Learning Logical Definitions from Relations". *Machine Learning*, vol. 5(3), 1990. Págs. 239-266.

[Quinlan, 91]
J. R. QUINLAN. "Knowledge Acquisition from Structured Data. Using Determinate Literals to Assist Search". *IEEE Expert*, vol. 6(6), Diciembre 1991. Págs. 32-37.

[Quinlan, 91b]
J.R. QUINLAN. "Determinate Literals in Inductive Logic Programming". En *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. Págs. 442-446.

[Quinlan, 93]
J. R. QUINLAN. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, The Morgan Kaufmann Series in Machine Learning, 1993.

[Quinlan, 95]
J. R. QUINLAN. “Learning First-Order Definitions of Functions”. En *URL ftp://*

ftp.cs.su.oz.au/pub/ml/q.foil.ps.

[Quinlan, 95b]
J. R. QUINLAN. “Past Tenses of Verbs and First-Order Learning”. En *URL ftp:// ftp.cs.su.oz.au/pub/ml/q.ai95.ps*

[Quinlan, 95c]
J. R. QUINLAN. “MDL and Categorical Theories (Continued)”. En *URL ftp:// ftp.cs.su.oz.au/pub/ml/q.ml95.ps*.

[Quinlan, 96]
J. R. QUINLAN. “Improved Use of Continuous Attributes in C4.5”. En *Journal of Artificial Intelligence Research*, vol. 4, 1996. Págs. 77-90. (En *URL http:// www.cs.washington.edu/research/jair/volume4/quinlan96a.ps*).

[Quinlan y Cameron-Jones, 93]
J. R. QUINLAN y R. M. CAMERON-JONES. "FOIL: a midterm report”. En *Proc. of the European Conference on Machine Learning*, Springer Verlag, 1993. Págs. 3-20. (En *URL ftp://ftp.cs.su.oz.au/pub/ml/q+cj.ecml93.ps*).

[Quinlan y Cameron-Jones, 94]
J. R. QUINLAN y R. M. CAMERON-JONES. “Living in a Closed World”. En *URL ftp://ftp.cs.su.oz.au/pub/ml/q+cj.closed.ps*.

[Quinlan y Cameron-Jones, 95]
J. R. QUINLAN y R. M. CAMERON-JONES. "Induction of Logic Programs: FOIL and Related Systems". En *New Generation Computing*, vol. 13, 1995. Págs. 287-312. (En *URL ftp://ftp.cs.su.oz.au/pub/ml/q+cj.ngc95.ps*).

[Quinlan y Cameron-Jones, 95b]
J. R. QUINLAN y R. M. CAMERON-JONES. “Oversearching and Layered Search in Empirical Learning”. En *URL ftp://ftp.cs.su.oz.au/pub/ml/q+cj.ijcai95.ps*.

[RAE, 94]
REAL ACADEMIA ESPAÑOLA. *Diccionario de la Lengua Española*. Ed. Espasa Calpe, Madrid, 1994.

[Rau, 93]
L. F. RAU. "Calculating Salience and Breadth of Knowledge". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 996-998.

[Richards y Mooney, 92]
B.L.RICHARDS y R.J.MOONEY. "Learning Relations by Pathfinding". *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, Julio 1992.

Págs. 50-55. (En *URL ftp://cs.utexas.edu/pub/mooney/papers/aaai92-relpath*).

[Römer et al. 95]
C. RÖMER, A. KANDEL y E. BACKER. "Fuzzy Partitions of the Sample Space and Fuzzy Parameter Hymotheses". En *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25(9), Septiembre 1995. Págs. 1314-1321.

[Salzberg, 95]
S. L. SALZBERG. "On Comparing Classifiers: A Critique of Current Research and Methods". *Technical Report JHU-95/06*, Dept. of Computer Science, Johns Hopkins University, Mayo 1995. Págs. 1-12. (En *URL http://www.cs.jhu.edu/salzberg/critique.ps*).

[Sammut y Banerji, 86]
C. SAMMUT y R. BANERJI. "Learning Concepts by Asking Questions". En *Machine Learning: An Artificial Intelligence Intelligence Approach*, vol. 2, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Tioga Publishing, Palo Alto, California, 1986. Págs. 167-191.

[Sandberg, 81]
G. SANDBERG. "A primer on relational data base concepts". *IBM Journal*, vol. 20(1), 1981. Págs. 23-40.

[Savnik, 93]
I. SAVNIK. “Bottom-up Induction of Functional Dependencies from Relations”. En *URL ftp://ftp-csd.ijs.si/Reports/CSD-TR-93-3.ps.gz*. (En *Proc. of Knowledge Discovery from Databases Workshop*, G. P. Shapiro ed., 1993). (En *Report CSD-93-3*, Computer Systems Department, Josef Stefan Institute, 1993).

[Schachte y Saab, 93]
P. SCHACHTE y G. SAAB. "Efficient Object-Oriented Programming in Prolog". En *URL http://www.cs.mu.oz.au/tr-db/mu_93_29.ps.gz*, 1993. Págs. 0-21.

[Sedgewick. 92]
R. SEDGEWICK. *Algorithms in C++*, Addison-Wesley Publishing Company, 1992.

[Serrano, 95]
D. SERRANO. *Sistema para Inducción de Conocimiento con Incertidumbre en Bases de Datos*, Proyecto Fin de Carrera, Dpto. Ing. Sistemas Telemáticos, E.T.S.I. Telecomunicación, Universidad Politécnica, Madrid, 1995.

[Shapiro, 87]
S. C. SHAPIRO. *Encyclopedia of Artificial Intelligence*. Ed. John Wiley & Sons, 1987.

[Shavlik et al., 90]
J. W. SHAVLIK, R. J. MOONEY y G. G. TOWELL. “Symbolic and Neural Learning Algorithms: An Experimental Comparison”. Computer Sciences Technical Report

955, University of Wisconsin, August 1990. En *URL ftp://ftp.cs.wisc.edu/tech-reports/reports/90/tr955.ps.Z*. (Appears in Machine Learning, vol. 6, 1991).

[Shekhar et al., 93]
S. SHEKHAR, B. HAMIDZADEH, A. KOHLI y M. COYLE. "Learning Transformation Rules for Semantic Query Optimization: A Data-Driven Approach". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 950-964.

[Shortland y Scarfe, 94]
R. SHORTLAND y R. SCARFE. "Data Mining applications in BT". En *BT Technology Journal*, vol. 12(4), Octubre 1994. Págs. 17-22.

[Silverschatz et al., 95]
A. SILBERSCHATZ, M. STONEBRAKER y J. ULLMAN. “Database Research: Achievements and Opportunities Into the 21st Century”. Report of an NSF Workshop on the Future of Database Systems Research, May 1995. En *URL http://db.stanford.edu/pub/ullman/logii.ps*.

[Silverstein y Pazzani, 91]
G. SILVERSTEIN y M. J. PAZZANI. "Relational clichés: Constraining constructive induction during relational learning". En *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. Págs. 203-207. (En *URL ftp://ftp.ics.uci.edu/pub/machine-learning-papers/publications/Silverstein-MLC91-Cliche.ps*).

[Simon, 84]
H. A. SIMON. "Why should machines learn?". *Machine Learning: An Artificial Intelligence Intelligence Approach*, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Tioga Publishing, Palo Alto, California, 1984. Págs. 25-37.

[Simon y Valdés-Pérez, 92]
H. A. SIMON y R. E. VALDÉS-PÉREZ. “The Evolution and Reporting of Discovery Systems”. En *URL http://www.cs.cmu.edu/afs/cs/uset/Valdes/Ftp/mdw92.pp.ps.*

[Simoudis, 96]
E. SIMOUDIS. “Reality Check for Data Mining”. En *IEEE Expert*, vol. 11(5), Oct. 1996. Págs. 26-33.

[Smyth y Goodman, 92]
P. SMYTH y R. M. GOODMAN. "An Information Theoretic Approach to Rule Induction from Databases". En *IEEE Transactions on knowledge and Data Engineering*, vol. 4(4), Agosto 1992. Págs. 301-316.

[Sommer et al., 93]
E. SOMMER, K. MORIK, J. M. ANDRE y M. USZYNSKI. "What online Machine Learning can do for Knowledge Acquisition- A case study". En *URL ftp://ftp.gmd.de/*

MachineLearning/GMD/papers/Sommer.etal93a.ps.Z, 1993. Págs. 1-27.

[Sudkamp y Hammell, 94]
T. SUDKAMP y R. J. HAMMELL. “Interpolation, Completion and Learning Fuzzy Rules”. En *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24 (2), Febrero 1994. Págs. 332 - 342.

[Thieme y Siebes, 93]
C. THIEME y A. SIEBES. "Schema Integration in Object-Oriented Databases". En *URL ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9320.ps.Z*. 1993. Págs. 1-30.

[Thieme y Siebes, 93b]
C. THIEME y A. SIEBES. "Schema Refinement and Schema Integration in Object-Oriented Databases". En *URL ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9354.ps.Z*. 1993. Págs. 1-19.

[Thornburg y Michalski, 86]
G. THORNBURG y R. S. MICHALSKI. "Machine Learning: Challenges of the Eighties". En *Machine Learning: An Artificial Intelligence Intelligence Approach*, vol. 2, R. S. Michalski, J. G. Carbonell y T. M. Mitchell, Morgan Kaufmann Publishers, Los Altos, California, 1986. Págs. 3-25.

[Thrun et al., 86]
S. B. THRUN et al. *The MONK's Problems: A Performance Comparison of Different Learning Algorithms*, Technical Report, Carnegie Mellon University, (CMU-CS-91-197), Diciembre 1991. Págs. 1-112.

[Uszynski, 86]
M. USZYNSKI. "Fuzzy Queries with Linguistic Quantifiers for Information Retrieval from Data Bases". En *URL ftp://tr-ftp.es.berkeley.edu/pub/tech-reports/csd/csd-87-333*, Julio 1986. Págs. 0-11.

[Utgoff, 89]
P. E. UTGOFF. "Incremental Induction of Decision Trees". En *Machine Learning*, vol. 4, 1989. Págs. 161-186.

[Velasco, 91]
J. R. VELASCO. *Arquitectura para Sistemas de Control Inteligente*, Tesis Doctoral, Dpto. Ing. Sistemas Telemáticos, E.T.S.I. Telecomunicación, Universidad Politécnica, Madrid, 1991.

[Viñas et al., 88]
J. VIÑAS, J. RIERA y J. QUEMADA. *Teoría de la información y codificación*. E.T.S.I. Telecomunicación, Universidad Politécnica, Madrid, 1988.

[Walker, 87]
M. G. WALKER. "How Feasible is Automated Discovery?". *IEEE Expert*, vol. 2(1),

Primavera 1987. Págs 69-82.

[Wang y Mendel, 92]
L. X. WANG y J. M. MENDEL. “Generating Fuzzy Rules by Learning through Examples”. En *IEEE Trans. on Systems, Man and Cybernetics*, vol. 22 (6), Noviembre 1992. Págs. 1414- 1427.

[Weiss et al., 87]
S. WEISS, R. GALEN y P. TADEPALLI. "Optimizing the Predictive Value of Diagnostic Decision Rules". *Proceedings of AAAI-87*, Seatle, Washington, 1987. Págs. 521-525.

[Wirth y O'Rorke, 91]
R. WIRTH y P. O'RORKE. "Constraints on Predicate Invention". En *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, IL, 1991. Págs. 457-461.

[Wirth y O'Rorke, 92]
R. WIRTH y P. O'RORKE. "Constraints for Predicate Invention". En *Inductive Logic Programming*, S. MUGGLETON, Academic Press, San Diego (CA), 1992. Págs. 299-318.

[Wong et al., 95]
S. K. M. WONG, C. J. BUTZ y Y. XIANG. “*A Method for Implementing a Probabilistic Model as a Relational Database”. En URL http://www.cs.uregina.ca/ ~butz/Papers/Implements.ps*. Dep. of Computer Science, Univ. Regina, Regina, Canada.

[Wong y Leung, 95]
M. L. WONG y K. S. LEUNG. “Inducing Logic Programs With Genetic Algorithms: The Genetic Logic Programming System”. En IEEE Expert, vol. 10(5), Octubre 1995. Págs. 68-76.

[Wu, 94]
XINDONG WU. “AI’94 Tutorial on Intelligent Learning Database Systems”. Seventh Australian Joint Conference on Artificial Intelligence (AI’94). En *URL ftp:// coral.cs.jcu.edu.au/pub/research/HCV/KDD.ps*, Noviembre 1994.

[Wuthrich, 95]
B. WUTHRICH. "Knowledge Discovery in Databases". Draft of a manuscript for a posgraduate course taught at the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. En *URL ftp://ftp.cs.ust.hk/techreport/95/tr95-04.ps.gz*, Febrero 1995. Págs 1-101.

[Yeh y Chen, 96]
MING-SHIOW YEH y SHYI-MING CHEN. “An Algorithm for Generatinf Fuzzy Rules from Relational Database Systems”. En *URL http://www.twnic.net/contest/*

winner/week3/Paper/sixth/31.html. National Chiao Tung University, Hsinchu.

[Yoon y Kerschberg, 93]
J. P. YOON y L. KERSCHBERG. "A Framework for Knowledge Discovery and Evolution in Databases". En *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), Diciembre 1993. Págs. 973-979.

[Zadeh, 65]
L. A. ZADEH. “Fuzzy Sets”. En *Information and Control*, vol. 8(3), 1965. Págs. 338-353.

[Zadeh, 73]
L. A. ZADEH. “Outline of a New Approach to the Analysis of Complex Systems and Decision Processes”. En *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3(1), Enero 1973. Págs. 28-44.

[Zadeh, 75]
L. A. ZADEH. “Fuzzy Logic and Approximate Reasoning”. En *Synthese*, vol. 30(1), 1975. Págs. 407-428.

[Zadeh, 84]
L. A. ZADEH. "Syllogistic Reasonig in Fuzzy Logic and its Applications to Reasoning with Dispositions". En *URL ftp://tt-ftp.cs.berkeley.edu/pub/tech-reports/cogsci/cogsci-84-16/all.ps*, Febrero 1984. Págs. 1-17.

[Zadeh, 96]
L. A. ZADEH. “The roles of fuzzy logic and soft computing in the conception, design and deployment of intelligent systems”. En BT Technology Journal, vol. 14(4), Oct. 1996. Págs. 32-36.

[Zelle et al., 94]
J. M. ZELLE, R. J. MOONEY y J. B. KONVISSER. “Combining Top-down and Bottom-up Techniques in Inductive Logic Programming”. En *URL ftp://ftp.su.utexas.edu/pub/mooney/papers/chill_ml_94.ps.Z.* (Appears in *Machine Learning: Proceedings of the Eleventh International Conference*, págs. 343-351, Morgan Kaufmann, 1994).